

ISSN 0868-6157
Совместное советско-американское предприятие «СОВАМИНКО»
КОМПЬЮТЕР ПРЕСС
КЛАВИАТУРА:
от А до Z
11'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

11'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

Дорогой читатель!

На какой вопрос ответить сегодня труднее всего? Видимо, на этот: что завтра останется от союза нерушимого республик свободных, который навеки сплотила великая Русь? Также неведомо, какие отношения будут в следующем году между бывшими очень братскими республиками, но уже известно, что например, Прибалтика полностью отказывается от всей российской печатной продукции. В том числе и от нашего журнала. Не исключено, что примеру прибалтов последуют и прочие суверенные государства.

Дорогой читатель! Нам будет очень грустно с тобой расставаться. И чтобы в будущем году нам не глотать слезы разлуки, – давай подпишемся на наш журнал и, как прежде будем ежемесячно встречаться и судачить о новостях компьютерного мира.

В номере использована графика М.К.Эшера

Тексты проверены системой “ОРФО”

Адрес редакции:
113093, г.Москва, аб.ящик 37
Факс: 200-22-89
Телефоны для справок:
491-01-53, 420-83-80.
E-mail:
postmaster@Computerpress.msk.su

Сдано в набор 15.10.91. Подписано к печати 30.10.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). №034. Тираж 100 000 экз. (1 завод–55 000). Заказ № 2503. Цена 3 р. 15 к.

Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18



Писать об устройствах типа Bernoulli Box, не упомянув при этом имени швейцарского ученого Даниила Бернулли, так же невозможно, как, рассказывая об истории авиации, не вспомнить братьев Райт. Хотя впрочем, если не к авиации, то к аэродинамике почетный член Петербургской Академии Наук Д. Бернулли имеет непосредственное отношение.

Из истории Bernoulli

Еще из курса средней школы известен закон течения идеальной жидкости или газа, описываемый уравнением Бернулли: давление на поверхность, создаваемое потоком движущейся жидкости или газа, зависит от скорости этого потока. Причем, чем быстрее движется газ (или жидкость), тем меньше давление на поверхность. Этот, на первый взгляд, достаточно скучный феномен уже давно имеет яркое практическое применение. Благодаря тому, что профиль несущей поверхности самолета выполнен определенным образом, скорость движения воздуха над верхней частью поверхности выше, чем под нижней. Результирующая разница давлений и обуславливает подъемную силу.

Специалисты американской фирмы Iomega нашли красивое решение для минимизации расстояния между магнитным слоем носителя и головкой записи/считывания, используя для регулирования этого расстояния известное соотношение Бернулли. Внешне новый носитель данных — Bernoulli Cartridge — выглядит как увеличенная до 5,25 дюймов обычная 3,5-дюймовая дискета. Движение воздуха в этой конструкции создается благодаря быстрому вращению диска. Неподвижный диск с магнитным носителем прогибается под тяжестью собственного веса, и, поскольку он расположен ниже головки, отдаляется от нее. При выборе оптимальной скорости вращения магнитный слой носителя и головку разделяет крошечная прослойка воздуха (три миллионных миллиметра). Головка "летит" над рабочей поверхностью носителя и, как следствие, отсутствует ее износ. При снижении количества оборотов, например, из-за отключения электропитания, расстояние между поверхностью магнитного носителя и универсальной головкой автоматически увеличивается.

Рассматриваемые ниже накопители, использующие эффект Бернулли (Removable Cartridge Drive, RCD), представляют уже второе поколение подобных устройств, используя в качестве сменного носителя — картриджа — теперь уже 5,25-дюймовые кассеты, вместо прежних 8-дюймовых. Это модели Bernoulli Box II, Bernoulli 20Z и Bernoulli II 44. Версии этих устройств имеют как встроенное (internal subsystem), так и внешнее исполнение (external subsystem). Первые две модели RCD позволяют использовать картриджи емкостью 20 Мбайт, последняя — 44 Мбайта.

Некоторые характеристики этих устройств приведены на рис.1. Следует пояснить, что устройство

Параметры	Устройства		
	Bernoulli Box II 44 M	Bernoulli 20Z	Bernoulli II 20 M
Форматированная емкость, Мб	44,5	21,4	21,4
Время выборки, мс	22	24	28
Скорость передачи данных, Мб/с	< 5,5	< 4,9	< 3,9
Чередование секторов	1:1	1:1	1:1
Время старт/стопа, с	5	7	7
Тип контроллера	SCSI	SCSI	SCSI
Буфер контроллера, Кб	32	8	8
Метод записи	RLL	RLL	RLL
Код коррекции	Рид-Соломон	48 бит ECC	48 бит ECC
Среднее время безотказной работы, лет	12	12	12

Рис.1.

Bernoulli 20Z является улучшенным вариантом Bernoulli Box II, это касается как скорости передачи данных, так и времени выборки. Причем владельцы накопителей Bernoulli Box II могут путем простой модернизации плат улучшить рабочие характеристики накопителей до значений параметров Bernoulli 20Z.

В комплект поставки, помимо одного или двух RCD, кабелей, документации и программ, входит плата Host-адаптера для компьютера (например, IBM PC). Она имеет половинную ширину (соответственно, использует 8-разрядную шину данных). В общем случае на плате находятся два десятка микросхем, 50-контактный разъем интерфейса SCSI, микропереключатели и поле для установки перемычек. Тип адаптера — PC2B/50 или PC2/50 — определяется наличием на плате микросхем ОЗУ (RAM), ПЗУ (ROM) и микросхемы управляющей логики, позволяющие выполнять загрузку с системного картриджа RCD. Но об этом чуть позже.

Микропереключатели на плате определяют область адресов адаптера для компьютера (340h...375h), номер используемого канала прямого доступа к памяти (DMA 1 или 3), способ ввода/вывода данных (программный или с использованием DMA), количество подсоединенных RCD (один или два). Перемычки на плате устанавливают начальный адрес ROM (в диапазоне C8000h...E4000h), при этом возможно использовать одно из 15 различных значений в указанном диапазоне. Это объясняется тем, что разные типы IBM-совместимых компьютеров могут использовать некоторые из этих областей памяти. Например, в компьютерах с процессором i8088, область памяти, начинающаяся с C8000h, обычно занята под ПЗУ BIOS жесткого диска. В некоторых PC/AT под этот BIOS занята память начиная с адреса E0000h.

Подсоединение двух RCD к Host-адаптеру производится по принципу "дэйзи-цепочки" (daisy-chaining) — Master (управляющий) и Slave (управляемый), то есть. Пассивное устройство управляется через контроллер активного.

Драйвер RCD.SYS, поставляемый фирмой Iomega в составе программного обеспечения, позволяет использовать накопители с версией MS DOS не ниже 3.0. Если Host-адаптер для RCD имеет тип PC2B/50, то имеется возможность выполнить загрузку компьютера с системного картриджа. Правда, здесь имеются и другие ограничения. Во-первых, известно, что POST BIOS позволяет выполнять загрузку DOS либо с устройства A:, либо с устройства C:, таким образом, чтобы стартовать с RCD, необходимо (иногда даже физически) отключить имеющийся винчестер, воспринимаемый обычно как устройство C:. Во-вторых, версии DOS ниже 4.0 поддерживают разделы жесткого диска размером не более 32 Мбайт. И хотя драйвер RCD.SYS позволяет работать с 44-Мбайтными картриджами под ранними версиями DOS, для того, чтобы загрузиться с такой кассеты, необходимо разбить ее на разделы в 32 Мбайта (загрузочный) и 12 Мбайт (увы! никак уже не используемый). Поскольку пять утилит для MS-DOS заменены для Iomega RCD собственными утилитами (рис.2.), форматирование картриджа или раздела производится утилитой RCD FORMAT с ключом /S (перенос системных файлов). На системном разделе кассеты, помимо файлов системы, должны присутствовать файлы RCD.SYS, CONFIG.SYS и AUTOEXEC.BAT. При этом файл CONFIG.SYS должен содержать минимум две строки

DEVICE = RCD.SYS
BUFFERS = 8 (конечно, можно и больше)

В файле AUTOEXEC.BAT рекомендуется установить команду VERIFY ON для проверки операций ввода/вывода.

При использовании RCD несомненный интерес представляет процедура резервного копирования диска (Backup), осуществляемая собственной утилитой RCD BACKUP. Эта утилита дает возможность использовать две разновидности этой операции: Usable Backup и Archival Backup. Метод Usable Backup эффективен, когда емкость винчестера меньше, чем емкость картриджа RCD, поскольку происходит непосредственное копирование файлов. В случае использования Archival Backup, перед последующим использованием файлов они должны быть восстановлены утилитой RCD RESTORE. Этот метод рекомендуется в случае, когда емкость используемого винчестера больше емкости одного картриджа RCD. Решение о том, какой из методов применить в конкретном случае, зависит и от количества дублируемых файлов. Если их суммарный размер не превосходит 44 Мбайт (для Bernoulli Box 44), то копирование файлов быстрее выполняется при использовании Usable Backup, в противном случае эффективнее окажется Archival Backup.

MS DOS утилиты	RCD утилиты
BACKUP	RCD BACKUP
DISKCOPY	RCD COPY
FDISK	RCD PARTITION
FORMAT	RCD FORMAT
RESTORE	RCD RESTORE

Рис.2.

Следует также отметить, что фирма Iomega обеспечивает совместимость своих изделий с сетевым программным обеспечением фирмы Novell. Тип используемого компьютера практически не накладывает ограничения на применение устройств типа Bernoulli Box (при использовании соответствующего адаптера). Это могут быть PC/XT, PC/XT286, PC/AT, PC/AT386, PS/2 (модели 60, 80), Compaq Portable 286, Compaq DeskPro, Tandy 3000, Unisys и т.д. Например, журнал Mikrocomputer Zeitschrift упоминает о 70 протестированных моделях компьютеров.

Особенности устройств типа Bernoulli Box, такие как емкость, скорость передачи данных (на уровне параметров жестких дисков), возможность смены кассеты носителя, высокая надежность хранения данных и практическая нечувствительность к механическим воздействиям, обуславливают и сферы их применения. Так, например, применение RCD в наших, зачастую коллективно-персональных "писишках", дает неплохой шанс сделать их действительно персональными. Конфиденциальность и сохранность ценных данных будут гарантированы. Если использовать RCD как устройство для загрузки DOS, то вероятность появления вирусов в Вашей системе сводится практически к нулю. Применение RCD как дополнения к винчестеру для хранения, например, больших графических массивов, требующих оперативного доступа, практически не имеет альтернатив (если исключить сменные жесткие диски). Фирма Iomega гарантирует долговечность хранения данных на кассетах в течение *5* лет.

С.Трофимов
А.Борзенко

Мы продолжаем публикацию статьи о безопасности компьютерных систем, начатую в прошлом номере. Когда журнал с первой частью статьи был уже в типографии, мы узнали о некоторых интересных событиях, имевших место в нашей стране летом этого года. Более подробно об этом вы можете прочитать в еженедельнике "Коммерсантъ" №40 (90) от 30 сентября-6 октября 1991 года, мы же лишь вкратце напомним вам суть дела. Случилось именно то, о чем КомпьютерПресс предупреждал уже давно, и наиболее настойчиво — в предыдущем номере. Во Внешэкономбанке СССР начальником отдела вычислительного центра было совершено крупное хищение валюты. Кража проводилась с помощью компьютерной программы, автором которой был сам преступник. Эта программа использовалась в банке уже 10 лет. Самое потрясающее во всей этой истории то, что никто происшедшему особенно не удивился, а главное — не обеспокоился. Соответствующий отклик это событие вызвало только в узких компьютерных кругах, занимающихся компьютерной безопасностью. Ни банковские работники, ни многочисленные брокерские конторы на событие почти не отреагировали. Они по-прежнему не понимают или не хотят понимать, что ходят по лезвию ножа. Вот уж, поистине, олимпийское спокойствие. Видно, нашего мужика ничем не прошибешь — он не крестится и после того, как грянул гром. И, между прочим, совершенно напрасно. Нет никаких сомнений, что компьютерная преступность, как и всякая другая преступность в нашей стране, будет расти. И мы к этому совершенно не готовы. У нас любой мало-мальски квалифицированный пользователь может залезть в компьютерную сеть, устроить там погром, получить конфиденциальную информацию и спокойно с достоинством удалиться. У нас ведь нет специалистов по компьютерной преступности! У нас нет соответствующих законов! Следователь по делу кражи в ВЭБ признался, что ничего не смыслит в компьютерах. Интересно, много ли они там нарасследуют? Ведь компьютерная программа могла переводить деньги не только на те счета, которые "засветились". Кто в этом сможет разобраться? Да, видно, чудеса в нашей стране еще только начинаются. Мы, как те глупцы, предпочитаем учиться на своих собственных ошибках, не признавая чужого опыта. Ну что ж, будем надеяться, что естественный отбор все же поможет продвинуться в бизнесе людям умным, а не только хитрым и пронырливым. Как говорится, что бог ни делает — все к лучшему.



Безопасность компьютерных систем

"По-настоящему безопасной можно считать лишь систему, которая выключена, замурована в бетонный корпус, заперта в помещении со свинцовыми стенами и охраняется вооруженным караулом, — но и в этом случае сомнения не оставляют меня".

Юджин Х. Спаффорд

2. Безопасность программ

Условимся, что под термином "программа" мы в равной степени будем понимать обычные программы пользователей; специальные программы, рассчитанные на нарушение безопасности системы; а также разнообразные системные утилиты и коммерческие прикладные программы, которые отличаются более высоким профессиональным уровнем разработки и, тем не менее, могут содержать отдельные недоработки, позволяющие захватчикам атаковать системы.

Программы могут порождать проблемы двух типов: они, во-первых, могут перехватывать и модифицировать данные в результате действий пользователя, который к этим данным не имеет доступа; и, во-вторых, используя упущения в защите компьютерных систем, программы могут или обеспечивать доступ к системе пользователям, не имеющим на это права, или блокировать доступ к системе законных пользователей.

К сожалению, количество возможных слабых точек, которые могут содержаться в той или иной программе, значительно превышает число известных технологий устранения экспозиций. Это обусловлено двумя причинами:

- во-первых, качество программы всегда не превышает квалификации ее разработчика: очевидные экспозиции и нарушения могут быть выявлены и устранены достаточно легко, однако, чем выше уровень подготовки программиста, тем более неявными (даже для него) становятся допускаемые им ошибки и тем более тщательно и надежно он способен скрыть умышленные механизмы, разработанные для нарушения безопасности системы;
- во-вторых, имеет место трудно разрешимый компромисс между эффективностью и удобством компьютерной системы в работе и степенью обеспечения в ней требований безопасности. Чем более высокие требования предъявляются к безопасности системы, тем большее количество ресурсов системы затрачивается на обеспечение этих требований, тем сильнее снижается производительность системы и увеличиваются сроки решения задач, наконец, тем неудобнее работать в данной системе пользователям. С другой стороны, чем больше ресурсов выделяется для решения текущих задач, тем меньше возможностей по обеспечению должного уровня безопасности.

В основном небезопасность программ состоит в том, что они могут быть использованы как средства получения критичной информации (данных), циркулирующей в системе, тем более, что данные в компьютерной системе обычно хранятся в виде, непонятном для большинства людей.

Впрочем, целью атаки могут быть и сами программы. Причин тому несколько:

1. В современном мире программы могут быть товаром, приносящим немалую прибыль, особенно тому, кто первый начнет тиражировать программу в коммерческих целях и оформит авторские права на нее. Поэтому опасность похищения почти готовой программы конкурентом — не такая уж надуманная возможность.
2. Программы могут становиться также объектом атаки, имеющей целью модифицировать эти программы некоторым образом, что позволило бы в будущем провести атаку на другие объекты системы. Особенно часто объектом атак такого рода становятся программы, реализующие функции защиты системы.

Рассмотрим несколько типов программ и приемы, которые наиболее часто используются для атак программ и данных.

2.1 Как залезть в программу

Как залезть в программу? Очень просто — через люк!

Люком называется не описанная в документации на программный продукт возможность работы с этим программным продуктом. Сущность использования люков состоит в том, что при выполнении пользователем некоторых не описанных в документации действий он получает доступ к возможностям и данным, которые в обычных условиях для него закрыты (в частности — выход в привилегированный режим).

Люки чаще всего являются результатом забывчивости разработчиков. В процессе разработки программы разработчики часто создают временные механизмы, облегчающие ведение отладки за счет прямого доступа к отлаживаемым частям продукта. Пусть, например, для начала работы с продуктом требуется выполнить некоторую последовательность действий, предусмотренных алгоритмом, — ввести пароль, установить значения некоторых переменных и т.п. При нормальной работе продукта эти действия имеют определенный смысл. Но во время отладки, когда разработчику необходимо тестировать некоторые внутренние части программы и волей-неволей приходится выполнять ту же операцию входа добрый десяток — а то и более — раз на дню, безобидные в общем-то правила, затрудняющие тем не менее доступ к отлаживаемым частям, начинают не на шутку раздражать. Что же делает программист? Правильно: в течение получаса он программирует некоторый дополнительный механизм, не предусмотренный изначальным алгоритмом программы, но позволяющий не выполнять надоедливых действий или выполнять их автоматически — например, при нажатии определенной клавиши (комбинации клавиш) или при вводе определенной последовательности символов. Все — люк готов!

По окончании отладки большинство люков убирается из программы; но люди есть люди — зачастую они забывают о существовании каких-то мелких “лючков“. Иногда, правда, разработчики сознательно оставляют люки в своих продуктах, особенно в ранних версиях, когда весьма вероятна возможность доработки продукта.

Одним из наиболее показательных примеров использования “забытых“ люков является, пожалуй, широко известный в компьютерном мире инцидент с вирусом Морриса. Одной из причин, обусловивших возможность распространения этого вируса, была ошибка разработчика программы электронной почты, входящей в состав одной из версий операционной системы UNIX, приведшая к появлению малозаметного лючка. Для вас, наверное, будет небесполезно знать, что американские специалисты оценивают ущерб, нанесенный в результате этого инцидента, более чем в 100 миллионов долларов.

Люки могут образовываться также в результате часто практикуемой технологии разработки программных продуктов “сверху вниз“. При этом программист приступает к написанию сразу управляющей программы, заменяя предполагаемые в будущем подпрограммы так называемыми “заглушками“ — группами команд, имитирующими или просто обозначающими место подсоединения будущих подпрограмм. В процессе разработки по мере готовности реальных подпрограмм эти подпрограммы заменяют соответствующие заглушки. В теории моментом завершения разработки конечной программы по такой технологии можно считать момент замены последней заглушки реальной подпрограммой (при условии, что все подпрограммы отлажены и синхронизированы).

В действительности дело обстоит несколько сложнее. Вся беда в том, что авторы часто оставляют заглушки в конечном программном продукте, передаваемом в эксплуатацию. Делают это порой неумышленно: например, на ранних стадиях разработки предполагалось наличие в конечном продукте некоторой подпрограммы, однако в процессе разработки выяснилось, что эта подпрограмма в силу каких-либо причин не нужна. Но заглушка-то на предполагавшемся месте подключения подпрограммы осталась! Более того, удалить заглушку, не заменяя ее подпрограммой, бывает весьма сложно. Все это может спровоцировать программиста на то, чтобы оставить такую заглушку на месте “до лучших времен“.

Возможен вариант, когда заглушки оставляются в конечной программе сознательно, в расчете на подключение в дальнейшем к работающей программе новых подпрограмм, реализующих некоторые новые возможности, либо предполагая возможное подключение к программе тестирующих средств для более точной настройки программы. Будет ли пользователь данной программы приобретать эти “новые возможности“, обратится ли он когда-нибудь к разработчику программы с просьбой более тонко настроить его детище — это еще под вопросом, а ведь заглушки-то стоят! Кто может дать гарантию, что в один прекрасный момент такой заглушкой кто-нибудь не воспользуется для подключения к программе совсем иной подпрограммы, работающей в интересах этого “кого-нибудь“, а не законного пользователя?

Наконец, еще одним распространенным источником люков является так называемый “неопределенный ввод“.

Допустим, некто предлагает вам программу, которой можно управлять с помощью определенных команд или даже просто путем ввода в ответ на запросы программы символов “Y“ (“Да“) или “N“ (“Нет“). А что произойдет, если вы в ответ на запрос введете, предположим, “А“ или вместо верной команды введете какую-либо абракадабру? Если программа написана хорошо, то такие случаи должны вызывать появление на экране сообщения типа “НЕВЕРНЫЙ ВВОД“ и повтор запроса.

Однако не так уж редка ситуация, когда программа создается неопытным программистом, исходящим из предположения, что пользователи будут работать с его

программой всегда корректно. В этом случае реакция программы на неопределенный ввод может быть в лучшем случае непредсказуемой (когда при повторном вводе той же неверной команды программа реагирует каждый раз по-разному); гораздо хуже, если программа в результате одинакового "неопределенного" ввода выполняет некоторые повторяющиеся действия — это дает потенциальному захватчику возможность планировать свои действия по нарушению безопасности,

В любом случае существует потенциальная возможность нарушения безопасности системы.

Неопределенный ввод — частная реализация ПРЕРЫВАНИЯ. То есть в общем случае захватчик может умышленно пойти на создание в системе некоторой нестандартной ситуации, позволившей бы ему выполнить необходимые действия. Например, он может искусственно вызвать аварийное завершение программы, работающей в привилегированном режиме, с тем, чтобы перехватить управление, оставшись в этом привилегированном режиме. Классический пример такого рода люков — ассемблерные команды SPIE и SPAE в ЕС ЭВМ, допускающие в определенных ситуациях перехват пользователем (или его программой) управления в режиме супервизора.

Борьба с возможностью прерывания (в терминологии безопасности) в конечном итоге выливается в предусмотрение при разработке программ комплекса механизмов, образующих так называемую "защиту от дурака". Смысл этой защиты состоит в том, чтобы гарантированно отсекать всякую вероятность обработки неопределенного ввода и разного рода нестандартных ситуаций (в частности, ошибок). Короче, качественная программа должна работать корректно и не приводить к появлению возможности нарушения безопасности компьютерной системы, даже если с самой программой работают некорректно — например, если за клавиатурой терминала случайно оказалась обезьяна.

Таким образом, люк (или люки) может присутствовать в программе ввиду того, что программист:

1) забыл удалить его;
2) умышленно оставил его в программе для обеспечения тестирования или выполнения оставшейся части отладки;
3) умышленно оставил его в программе в интересах облегчения окончательной сборки конечного программного продукта;
4) умышленно оставил его в программе с тем, чтобы иметь скрытое средство доступа к программе уже после того, как она вошла в состав конечного продукта.

В первом случае люк — неумышленная, но серьезная брешь в безопасности системы. Во втором и третьем случаях люк — серьезная экспозиция безопасности системы. Наконец, в последнем случае люк — первый шаг к атаке системы.

В любом случае люки — это возможность получить управление вашей системой в обход защиты.

2.2 "Бойтесь данайцев, дары приносящих!"

Существуют программы, реализующие, помимо функций, описанных в документации, и некоторые другие функции, в документации не описанные. Такие программы называются троянскими конями.

Например, вы запускаете какую-то нужную вам программу — графический пакет, расчетную задачу или игру, наконец, — а через некоторое время обнаруживаете, что в вашей директории исчезли все файлы. Поскольку кроме запуска известной вам программы вы ничего другого не делали, вполне вероятно, что вы заподозрите что-то неладное и решите запустить ту же программу в другой (лучше специально для этого созданной) директории. Если эффект повторится — вы имеете дело с "троянским конем" в компьютерном исполнении.

Далеко не всегда обнаружить троянского коня бывает так просто — вероятность обнаружения тем больше, чем очевиднее результаты действия троянского коня. А если задачей троянского коня является не удаление ваших файлов, а изменение их защиты? Явно вы этого не увидите, но каково будет ваше удивление, когда через некоторое время выяснится, что конфиденциальные данные, содержавшиеся в одном из ваших файлов (например, проект важного документа), стали известны всем. Проверяете наличие файла — на месте, проверяете защиту файла — а оказывается, его может читать любой!

Однако такие фиксированные результаты представляют очень опасную для троянского коня угрозу разоблачения его деятельности и, соответственно, обнаружения. Более сложный троянский конь может быть запрограммирован таким образом, чтобы по изменении защиты файлов (в нашем примере) подавать захватчику (лицу, заинтересованному в срабатывании троянского коня) некоторый условный сигнал о доступности файлов; после выдачи сигнала троянский конь некоторое время выжидает, а затем возвращает защиту файлов в исходное состояние.

Такой алгоритм позволяет захватчику в течение некоторого времени делать с вашими файлами все, что угодно, и никаких следов атаки после этого не остается. Вы же остаетесь в полном счастливом неведении, т.е. с носом.

2.3 Какая неожиданность!

"Логической бомбой" обычно называют программу или даже участок кода в программе, реализующий некоторую функцию при выполнении определенного условия.

Я понимаю, что такое определение достаточно расплывчато, поэтому попытаюсь его пояснить.

Прежде всего "логическая бомба" отличается от других программ тем, что, "взрываясь", она реализует функцию, неожиданную или, хуже того, нежеланную для пользователя. Например, удаляет некоторые данные или разрушает некоторые системные структуры

2 Зак. 2503

данных. Впрочем, функция логической бомбы может быть и более безобидной: например вывод на экран терминала или на принтер некоторого сообщения шутливого или оскорбительного характера.

Бомбы, существующие в виде отдельных программ, обычно имеют малопонятные или, наоборот, любопытные имена. Ничего не подозревающий пользователь, получивший подобный "сюрприз", каким-либо образом, естественно, пробует запустить программу, чтобы понять, что она делает — вот тут-то и начинается самое интересное! Классическим примером бомбы является программа, распространенная в американских компьютерных сетях под названием Rockvideo. После ее запуска на экране дисплея можно увидеть мультипликационные картинки с американской рок-певицей Мадонной, причем показ завершается выдачей сообщения следующего содержания: "Только идиот использует свой компьютер для того, чтобы рассматривать видеозвезд!". Во время демонстрации бомба удаляет себя, но заодно удаляет и все файлы на доступных для нее дисках.

Есть также основания предполагать, что не слишком широко известный инцидент с остановкой главного конвейера на Горьковском автозаводе был также не чем иным, как "взрывом" "логической бомбы".

Естественно, поняв, что представляет собой программа-"подарок", пользователь удаляет ее, но это зачастую уже не имеет значения: бомба-то взорвалась. А уж разрушения в ваши данные она может внести самые сокрушительные: это определяется только уровнем подготовки и изощренностью автора бомбы.

Более сложным является случай, когда программа, осуществляющая некоторую законную обработку, содержит в себе участок кода, который срабатывает при выполнении в системе некоторых условий, например при наступлении определенной даты или при обнаружении файла с определенным именем. Такие вкрапления в тело обычных программ также можно считать "логической бомбой".

Мировая компьютерная общественность достаточно хорошо знакома с логическими бомбами. Логическая бомба является одним из излюбленных способов мести программистов компаниям, которые их уволили или чем-либо обидели. При этом чаще всего срабатывание бомбы ставится в зависимость от установки в системе даты — так называемые "часовые" бомбы. Это очень удобно: допустим, программист знает, что его уволят 1 марта; в таком случае он может установить "часовую" бомбу на взрыв, допустим, 6 июля или даже на Рождество, когда сам он будет уже вне пределов досягаемости для пострадавшей компании*.

Ориентируя бомбу на установку определенной даты, можно также закладывать бомбы впрок: например, можно определить, что бомба должна сработать 30 февраля. Поскольку такой даты не существует, бомба не взорвется до тех пор, пока знающий о ней злоумышленник преднамеренно (или ни о чем не догадывающийся оператор системы — по ошибке) не установит системные часы в соответствующее положение.

* В этом отношении интересна высказанная одним из администраторов систем мысль, что при увольнении системного программиста будет лучше, если глава фирмы проводит его до дверей офиса, вежливо попрощается и подаст пальто.

2.4 Колбаса и компьютер

Речь пойдет о биче банковских компьютерных систем — атаке "салями".

Чтобы понять смысл такой атаки, полезно вспомнить технологию изготовления известного сорта колбасы, которая создается путем соединения в общее целое множества мелких кусочков мяса. Получается достаточно вкусно.

Теперь рассмотрим смысл компьютерной атаки. В банковских системах ежедневно производятся тысячи операций, связанных с безналичными расчетами, переводами сумм, отчислениями и т.д. Во многих из таких операций возникает проблема вычисления различного рода долей от некоторой величины. Вычислить необходимый процент можно с разной степенью точности, и поэтому еще при разработке систем устанавливается правило округления (или усечения), используемое при выполнении всех операций.

Классическим примером атаки салями является, возможно, вымышленная история о вычислении доли в 6,5% от 102,87 долл. для 31 дня. Несложные вычисления дают:

$$31 / 365 * .065 * 102.87 = 0.5495726 \text{ долл.}$$

С точки зрения банка, различающего в качестве самой маленькой величины только целый цент, эта величина не имеет смысла, также как вряд ли кто-нибудь из нас сможет доступно объяснить, что такое "полкопейки". Поэтому в банках принято правило, по которому величины, превышающие половину цента, округляются с избытком (или просто округляются) до целого цента, а величины менее половины цента просто отбрасываются.

Вся хитрость состоит в том, как запрограммировать обработку отбрасываемых долей и округлений. Можно, конечно, просто удалять несущественные величины. Но можно и не удалять эти величины, а накапливать их постепенно на некоем специальном счете. Там полцента, тут полцента... — а в сумме? Как свидетельствует практика, сумма, составленная буквально из ничего, за пару лет эксплуатации "хитрой" программы в среднем по размеру банке, может исчисляться тысячами долларов.

Можно сказать, что атака салями — компьютерная реализация известной поговорки "С миру по нитке — голому рубаха".

Компьютерные вычисления по природе склонны к мелким погрешностям, особенно при одновременной обработке больших и маленьких величин. С другой стороны, для различных практических нужд человеку требуется различная точность вычислений. Расплачи-

ваясь, например, в такси, вы можете позволить себе поправку счета на размер "чаевых", но рассчитывая орбиту искусственного спутника, вы будете заинтересованы в вычислении всех величин с точностью "до энного знака после запятой". Однако обработка дробных величин на ЭВМ связана с достаточно сложными вычислениями. Именно ввиду этой изначальной сложности работы с дробными величинами обнаружить в программе код, реализующий атаку салями, бывает весьма непросто. А это в свою очередь объясняет регулярное появление сообщений о все новых и новых атаках этого типа.

2.5 Скрытые каналы

Теперь поговорим о программах, передающих информацию лицам, которые в обычных условиях эту информацию получать не должны. Называются такие необычные способы извлечения информации скрытыми каналами.

В тех системах, где ведется обработка критичной информации, программист не должен иметь доступа к обрабатываемым программой данным после начала эксплуатации этой программы. Например, банковский программист не должен иметь доступа к именам или счетам вкладчиков и клиентов, как не должен знать порядка обслуживания клиентов. В процессе отладки программы разработчику допускается предоставление ограниченного объема реальных данных для проверки работоспособности программы, но предоставление реальных данных после сдачи программы в эксплуатацию не имеет оправдания.

Из факта обладания некоторой служебной информацией можно извлечь достаточно немалую выгоду, хотя бы элементарно продав эту информацию (например, список клиентов) конкурирующей фирме. Поэтому нетрудно предположить, что кто-либо из сотрудников будет заинтересован в том, чтобы иметь возможность такую информацию получать.

Достаточно квалифицированный программист всегда может найти способ скрытой передачи информации; при этом программа, предназначенная для создания самых безобидных отчетов может быть немного сложнее, чем того требует задача. Для скрытой передачи информации можно с успехом использовать различные элементы формата "безобидных" отчетов, например, разную длину строк, пропуски между строками, наличие или отсутствие служебных заголовков, управляемый вывод незначащих цифр в выводимых величинах, количество пробелов или других символов в определенных местах отчета и т.д.

Например, определенную информацию может нести появление в отчете вместо служебного заголовка "TOTAL" другого заголовка — "TOTALS". Разница всего в один символ, но именно его наличие или отсутствие и может быть сигналом осведомленному о канале захватчику (например, о том, что в системе появился файл с определенным именем). В данном случае имеет место 1-битовый скрытый канал, так как за один раз передается один бит информации (наличие или отсутствие символа "S").

Однако возможна ситуация, когда захватчик не будет иметь доступа ко всем отчетам, создаваемым "хитрой" программой, и таким образом он будет лишен возможности пользоваться своим скрытым каналом непосредственно.

В таком случае программист может обеспечить, например, вызов в определенных ситуациях более безопасной программой, непосредственно обрабатывающей интересующие захватчика данные, менее безопасной программы, которой и будут переданы критичные данные. А уж вызванная программа может создавать замаскированный, внешне "безобидный" отчет, доступный для просмотра захватчику.

Еще более сложным может быть случай, когда захватчик не имеет возможности просматривать замаскированные отчеты или обеспечивать вызов подпрограмм-шпионов, но имеет возможность доступа к компьютеру во время работы интересующей его программы. В такой ситуации возможно создание специального массива данных непосредственно в оперативной памяти компьютера, куда программа, содержащая скрытый канал, будет пересылать критичную информацию. Причем, опять же, информация может быть представлена в виде служебных символов, старт-стопных сигналов для магнитных лент, символов управления консолью и т.д.

Как видно из приведенных примеров, для получения относительно небольших объемов информации захватчик вынужден проделать достаточно большую работу. Поэтому скрытые каналы наиболее применимы в ситуациях, когда захватчика интересует даже не содержание информации, а, допустим, факт ее наличия (например, наличие в банке расчетного счета с определенным номером). Можно предположить и более изощренную ситуацию: с помощью скрытого канала захватчик получит сигнал о появлении в системе файла с определенным именем, что в свою очередь служит признаком работы в системе некоторого процесса, позволяющего провести атаку иного типа.

2.6 Жадность — порок!

Большинство методов нарушения безопасности направлены на то, чтобы получить доступ к данным, недопускаемый системой в нормальных условиях. Однако не менее интересным для захватчиков является доступ к управлению самой компьютерной системой или изменение ее качественных характеристик. Это может потребоваться для того, чтобы явно использовать компьютерную систему в своих целях (хотя бы для бесплатного решения своих задач) либо просто заблокировать систему, сделав ее недоступной другим пользователям. Такой вид нарушения безопасности системы называется "отказом в обслуживании" или "отказом от пользы".

Некоторые компьютеры, особенно в исследовательских центрах, имеют так называемые "фоновые" за-

дачи, постоянно решаемые, но с очень маленьким приоритетом. Обычно это задачи, требующие для решения очень большого машинного времени, но не особенно срочные (типа вычисления числа е или Пи с очень большой степенью точности). Низкий приоритет делает возможным решение этих задач только при условии, что в системе нет задач с более высоким приоритетом. Однако по недосмотру либо в результате ошибки, либо умышленно приоритет такой "фоновой" задачи может быть существенно повышен: в этом случае бывшая "тихоходная" задача захватывает процессор в монопольное использование, блокируя выполнение всех других вычислений.

Такого рода программы, захватывающие некоторый ресурс в монопольное использование, называются жадными программами. Ведь чтобы остановить выполнение какого-либо задания, вовсе не обязательно захватывать именно процессор.

Например, если известно, что на некотором этапе задача, которую нужно заблокировать, должна выполнять вывод на печать некоторого отчета, можно заблокировать доступ к принтеру. В таком случае оператор системы будет видеть, что обработка заданий проходит "нормально" — процессор-то не заблокирован и другие задачи нормально решаются — и так до тех пор, пока не прибежит разгневанный хозяин заблокированной задачи в сопровождении администратора системы.

Изобрести подходящую "долгоиграющую" задачу для того, чтобы намертво заблокировать систему, несложно — достаточно запрограммировать в обычной программе явный или неявный бесконечный цикл.

В явном виде бесконечный цикл можно определить, поставив условием окончания цикла условие, которое никогда не будет выполнено. Ну, например, запрограммировать выполнение цикла, пока 1 больше 0. Да, единица всегда будет больше нуля! Таким образом, цикл будет выполняться хоть до второго пришествия.

В неявном виде цикл можно организовать с помощью двух команд перехода, которые передают управление друг другу. Это что-то вроде лемовских "сепулек" и "сепулькарий".

Бороться с бесконечными циклами можно с помощью имеющейся в большинстве систем возможности устанавливать предельное количество машинного времени для решения данной задачи. В таком случае по исчерпании задачей выделенного лимита времени, система автоматически останавливает задачу. Однако, во-первых, установленный лимит можно обойти или постоянно изменять, а во-вторых, сославшись на длительность решения задачи, можно потребовать от администратора системы вообще снять какие-либо временные ограничения для вашей задачи.

Кроме того, существует еще одна тонкость в исчислении машинного времени. Дело в том, что машинное время измеряется в единицах времени, затраченного центральным процессором на выполнение данной задачи. Таким образом, время ожидания задачей некоторого события — например окончания операции ввода/вывода — не учитывается. В некоторых системах, если задача запрашивает выполнение операции ввода/вывода, то на время ожидания этой задачей завершения запрошенной операции система "выключает" счетчик времени процессора для данной задачи. Но можно ведь создать программу ввода/вывода с бесконечным циклом! В таком случае остановится и внешняя программа, вызвавшая подпрограмму-ловушку.

Можно упомянуть также о знакомой программистам возможности вхождения двух процессов в клинч.

Допустим, в системе решаются две задачи — А и Б. Предположим также, что они имеют следующие алгоритмические участки:

задача А	задача Б
...	...
захватить ресурс 1	захватить ресурс 2
захватить ресурс 2	захватить ресурс 1
освободить ресурс 1	освободить ресурс 2
...	...

Теперь попытаемся представить, что произойдет, если обе задачи выйдут на выполнение этих участков одновременно. При этом задача А захватит в использование ресурс 1 (допустим, принтер); задача Б в то же время захватит ресурс 2 (предположим, плоттер). Далее задача А попробует захватить ресурс 2, но он-то уже захвачен задачей Б! Аналогично и с задачей Б, которой требуется захватить ресурс 1, уже захваченный задачей А.

Казалось бы, чего проще — пускай одна из задач освободит ресурс — и конфликт разрешится. Но в том-то весь секрет, что освобождение ресурса в обеих задачах зависит от выполнения захвата второго (в каждой задаче своего) ресурса, т.е. не выполнив захвата, задача не освободит другого ресурса.

Такое безвыходное положение, весьма похожее на ситуацию с двумя баранами на одном мосту из известной детской сказки, и называется клинчем.

Клинч может возникать случайно, в силу неудачно сложившихся обстоятельств; может возникнуть в результате низкого качества программирования; но нет, однако, принципиальной невозможности провокации неким злоумышленником ситуации, чреватой вхождением в клинч.

В большинстве современных систем существуют специальные средства разрешения такого рода конфликтов. Более того, в настоящее время уже при проектировании систем предусматривается наличие встроенных механизмов, нацеленных на выявление и предотвращение угрожающих ситуаций. И тем не менее угроза клинча остается.

Насколько опасной может быть блокировка компьютерной системы?

Прежде всего, спросите военных, насколько опасной может быть блокировка компьютера, управляющего различного рода системами раннего обнаружения или той же противовоздушной обороны.

Можно поинтересоваться у врачей, насколько серьезными могут быть последствия блокировки компьюте-

ров, осуществляющих мониторинг больных в реанимационном отделении.

Уже из этих примеров становится очевидным, что "отказ от обслуживания" чрезвычайно опасен для так называемых систем реального времени — систем, управляющих некоторыми технологическими процессами, осуществляющих различного рода синхронизацию и т.д.

2.7 Компьютерные вирусы

Компьютерные вирусы — тема для отдельного очень интересного и очень длинного разговора.

Серьезного отношения компьютерные вирусы потребовали к себе относительно недавно; настолько недавно, что американский институт стандартов до сих пор не дал четкого определения компьютерного вируса, благодаря чему в трудах и работах специалистов-компьютерщиков царит полнейший плюрализм в трактовке этого термина.

В последнее время распространилось следующее определение:

КОМПЬЮТЕРНЫЙ ВИРУС — набор команд (программных или иных), который производит и распространяет свои копии в компьютерных системах и/или компьютерных сетях и преднамеренно выполняет некоторые действия, нежелательные для законных пользователей систем.

Рассмотрим подробнее ключевые элементы определения, чтобы лучше понять суть явления.

Компьютерный вирус — набор команд. Команды, составляющие тело вируса, могут иметь самую различную природу. Сюда могут входить команды какого-либо языка(-ов) программирования (наиболее распространенный случай), микропрограммные инструкции, управляющие символы и комбинации в телекоммуникационных сообщениях, различного рода параметры, а также команды языка управления заданиями. Таким образом, перспективное каноническое определение вируса не должно ограничивать среду существования вируса.

Компьютерный вирус распространяется. Каноническое определение вируса не должно также ограничивать способы распространения вируса. Важно то, что вирус может репродуцировать себя в одной или нескольких компьютерных системах**.

Существенно также то, что вирус в принципе может распространять набор команд, отличающийся по форме или содержанию от оригинала. Короче, вирус со временем может эволюционировать.

Компьютерный вирус выполняет нежелательные действия. Попадая тем или иным способом в вашу систему, вирус самокопирует себя в различные места памяти системы, а затем (либо одновременно с этим) производит в вашей системе изменения, в лучшем случае не приводящие к катастрофическим для вас последствиям (например, высвечивание на экране терминала некоторого сообщения), а в худшем — делающие вашу систему неработоспособной. Заметим, что большинство вирусов изначально небезобидны.

Так как вирус самостоятельно обеспечивает свое размножение и распространение, вам, в случае обнаружения вами вируса, необходимо проверить всю вашу систему, уничтожая копии вируса. Если вам удалось уничтожить все копии вируса, вы можете сказать, что вылечили вашу систему; в противном случае уцелевшие копии снова саморазмножатся и все неприятности повторятся сначала.

Своим названием компьютерные вирусы обязаны определенному сходству с вирусами естественными: способности к саморазмножению; высокой скорости распространения; избирательности поражаемых систем (каждый вирус поражает только определенные системы или однородные группы систем); способности "заражать" еще незараженные системы; трудности борьбы с вирусами и т.д. В последнее время к этим особенностям, характерным для вирусов компьютерных и естественных, можно добавить еще и постоянно увеличивающуюся быстроту появления модификаций и новых поколений вирусов. Только если в случае вирусов естественных эту скорость можно объяснить могуществом и изобретательностью природы, то вирусы компьютерные скоростью возникновения новых штаммов обязаны исключительно недосмотру или бредовым идеям людей определенного склада.

Приведем пример подпрограммы-вируса. Она состоит из псевдокоманд DOS и подпрограмм — маленьких внутренних программ (составляющие их инструкции хранятся отдельно от главной программы), выполняющих некоторые специальные функции всякий раз, когда к ним обращаются.

```
this: = findfile
LOAD(this)
loc: = search(this)
insert(loc)
STORE(this)
```

Подпрограмма под названием findfile обращается к каталогу выполняемых файлов или программ на диске, берет произвольное имя файла и присваивает имя этого файла переменной this (этот).

В следующей строке программы используется псевдокоманда DOS LOAD (загрузить), с помощью которой файл помещается в оперативную память компьютера.

Другая подпрограмма под названием search (поиск) просматривает только что загруженную программу в поисках инструкции, которая могла бы послужить подходящим местом, куда можно занести вирус. Когда процедура search находит такую инструкцию, она

** Кстати, именно способность саморазмножаться отличает программу-вирус от логической бомбы.

определяет соответствующий номер строки и присваивает его в качестве значения переменной loc.

Теперь все готово для того, чтобы подпрограмма-вирус могла проникнуть в произвольно выбранную из каталога программу. Подпрограмма insert (вставить) заменяет выбранную инструкцию другой (например такой, как вызов подпрограммы). Замененная инструкция передает управление блоку команд, составляющих главное тело подпрограммы-вируса, которая присоединяется к концу программы. Затем к концу добавленной подпрограммы присоединяется инструкция, возвращающая управление "зараженной" программе, на инструкцию, следующую за вставленной.

Таким образом, когда выполняется подпрограмма-вирус, выполняется также и подмененная инструкция зараженной программы. Исходная программа работает так, будто ничего особенного не произошло. Однако, на самом деле, подпрограмма-вирус воспользовалась каким-то мгновением, чтобы захватить власть над средствами операционной системы и прицепить свою копию к еще одной программе, хранящейся на диске.

Приведенный пример демонстрирует лишь один из приемов, используемых авторами вирусов. В настоящее время специалисты выделили и другие приемы, отличающиеся друг от друга идеями и изощренностью выполнения.

В широком потоке литературы по проблеме вирусов, хлынувшей на нас в последнее время, приводится множество классификаций и описаний вирусов. Так что интересующихся этой проблемой всерьез я с чувством глубокого облегчения отсылаю к специальным публикациям, большей частью, правда, далеких от совершенства и законченности.

2.8 Как возникают угрозы

Все рассмотренные выше виды опасных программных приемов иногда обозначают единым термином — "электронные ловушки".

Заслуживает внимания вопрос о том, как эти ловушки появляются.

В простейшем случае программист с самого начала разработки программы предусматривает наличие в ней какой-либо электронной ловушки, предположим троянского коня. Скомпилировав программу, он отдает свое изделие для использования в вычислительный центр или другому пользователю, не предоставляя при этом исходного кода программы. Анализировать двоичный код программы — дело весьма непростое, на несколько порядков более сложное, чем анализ текста программы на одном из языков программирования (хотя и исходный текст можно основательно запутать). Поэтому весьма велика вероятность того, что программа сразу поступит в эксплуатацию, чего, собственно, захватчику и нужно.

Зная о такой нехитрой технике обмана, многие центры принимают для обработки лишь исходные тексты программ. При этом компиляция и эксплуатация программы происходит в самом вычислительном центре, но только после анализа на наличие в программе электронных ловушек. Конечно, непременным условием для установления такой дисциплины является наличие в составе сотрудников центра высококлассного программиста, способного выполнить анализ исходного кода программы.

Еще раз отмечу, что поиск электронной ловушки по исходному тексту — очень непростая задача. Не стоит считать захватчика наивным: можно не сомневаться, что он предпримет все, чтобы тщательно замаскировать опасный код. Во всяком случае не ждите, что текст ловушки будет выделен комментарием типа: "Здесь приготовлен сюрприз!".

Существует много способов затруднить чтение программы. Например, можно отбросить всякие правила структурного программирования и поступить прямо противоположным образом — начать активно использовать в программе переходы из одной части в другую. Такие перескоки по тексту очень утомляют при чтении, поскольку трудно при этом не сбиться и не потерять смысл алгоритма.

Далее, существует немало способов вызова подпрограмм, одна из которых и может оказаться ловушкой. Вызов может осуществляться одной командой — попробуйте выделить эту команду среди десятков, а то и сотен других команд. Кроме того, вызов может быть и неявным: например, путем прямого занесения в счетчик команд адреса точки входа ловушки. В других системах это можно сделать путем установки указателя системного стека на соответствующую команду и т.д. Степень изощренности зависит от квалифицированности захватчика.

Если все же ловушка не выделена в некоторую самостоятельную подпрограмму, можно рассредоточить код ловушки по всему тексту программы. Код ловушки разбивается на пары команд, причем вторая команда каждой пары — команда перехода на следующую пару.

Еще пример? Можно запрограммировать ловушку таким образом, чтобы она использовала участки кода внешней программы, но в определенной последовательности и с добавлением небольших серий команд, разбросанных по всему тексту программы. Сами по себе эти добавочные команды могут выполнять внешне невинные, чаще всего бесполезные, с точки зрения анализирующего специалиста, действия. Легко ли разобраться в таком месиве команд? Как различить, где тело полезной программы, а где — раковая опухоль ловушки?

Кроме того, следует иметь в виду, что захватчик может внедрить электронную ловушку в уже проверенный исходный код программы, находящийся внутри центра. Это лишний раз подтверждает необходимость средств ограничения доступа пользователей к различным ресурсам системы, в частности — к текстам программ.

К сожалению, злоумышленное внесение кодов электронных ловушек в исходный текст программ — не единственный способ внедрить ловушку. Захватчиком

может оказаться высококлассный специалист, способный ввести код ловушки непосредственно в двоичный код программ (а многие вирусы, например, сами обеспечивают внедрение своих копий в исполняемые образы программ). Работа с двоичным исполняемым кодом программы требует весьма высокого уровня подготовки и надлежащих навыков в работе со специальными программными средствами — это своего рода “высший пилотаж“ программирования. Модификация двоичного кода может быть выполнена либо во время хранения программы на магнитном носителе в промежутках между ее вызовами, либо даже во время выполнения непосредственно в памяти машины.

Следует понимать, что в чистом виде описанные выше приемы встречаются достаточно редко. Гораздо чаще в ходе атаки используются отдельные элементы разных приемов. Например, “логическая бомба“ является фактически другой стороной “троянского коня“: ведь по сути внешняя программа, содержащая “логическую бомбу“, является ничем иным, как “троянским конем“. Вирусы вообще являются квинтэссенцией всевозможных методов нарушения безопасности. Иногда складывается впечатление, что все остальные способы атаки появились только для того, чтобы обеспечить возникновение компьютерных вирусов. Посудите сами: одним из самых частых и излюбленных способов распространения вирусов является метод “троянского коня“. От “логической бомбы“ вирусы отличаются только возможностью размножаться и обеспечивать свой запуск — так что многие вирусы можно считать особой формой “логических бомб“. Для атаки системы вирусы активно используют разного рода “люки“. Реализовывать вирусы могут самые разнообразные пакости, в том числе и атаку салями. Кроме того, успех атаки одного вида часто способствует снижению “иммунитета“ системы, создает благоприятную среду для успеха атак других видов. Захватчики это знают и активно используют указанное обстоятельство.

Все вышесказанное имеет целью подтолкнуть вас к закономерному и единственно верному выводу. Бороться даже с частным видом нарушений безопасности — электронными ловушками — как и с любыми другими видами нарушений, можно, только используя в комплексе все физические, логические и организационные средства и методы обеспечения безопасности.

3. Безопасность в операционных системах

Операционная система — “душа“ любой компьютерной системы — в принципе является очень большой и сложной программой. Так что многие слабости программ, о которых шла речь выше, присущи и операционным системам.

Однако есть ряд качеств, которые заставляют выделить вопросы обеспечения безопасности операционных систем в особую категорию.

Прежде всего операционная система часто содержит ряд встроенных механизмов, прямо или косвенно влияющих на безопасность всех программ и данных, работающих и обрабатываемых в среде данной операционной системы.

Помимо этого, размер и сложность операционной системы как программы делает качественно более сложной атаку на поражение операционной системы.

Чтобы испортить настроение администратору системы, вовсе не обязательно специально разрабатывать одну из описанных выше электронных ловушек. Нанести серьезный ущерб системе — и тем самым нарушить ее безопасность — можно и с помощью самых обычных программ или утилит. Например, удалить важный для кого-то файл можно с помощью одной команды операционной системы. Можно вообще отформатировать все доступные носители с помощью стандартной системной утилиты, уничтожив тем самым всю хранившуюся на них информацию и даже саму систему. Прочитать данные — те, которые вам читать не положено, — легче всего с помощью текстового редактора.

Таким образом, мы подошли к одной из центральных проблем в обеспечении безопасности: проблеме несанкционированного (неавторизованного) доступа и способам его предотвращения. Вся проблема заключается в обеспечении такого порядка работы, при котором систему мог бы использовать только тот, кому ее разрешено использовать; чтобы каждый законный пользователь работал только со “своими“ данными и не мог исказить, прочитать или удалить из системы данные, принадлежащие другому пользователю (если на то нет согласия хозяина); чтобы каждый законный пользователь мог выполнять только те операции, которые ему разрешено выполнять администратором системы.

В самом деле, вполне естественным является стремление не допустить работы в системе посторонних лиц: вы же не пускаете случайных прохожих в свой дом. Не менее естественно желание пользователей иметь гарантии, что их личные данные никто по ошибке или с умыслом не удалит, не исказит, не прочитает и т.д.

Кроме того, имеет прямой смысл предоставлять каждому новому пользователю для начала только самый необходимый набор средств, без которых он вообще работать не сможет — хотя бы в целях его же собственной безопасности. Вы же не даете ребенку молоток, справедливо опасаясь, что он попробует им шарахнуть по телевизору (или уж, по крайней мере, следите за ним). Начинающий пользователь для системы представляет не меньшую угрозу, чем ребенок для телевизора. По незнанию, неопытности или неосторожности пользователь может сотворить самые невероятные вещи: от простого удаления собственных данных (ну, с этим-то сталкивался практически каждый, кто имел дело с компьютером) до приведения в неработоспособное состояние всей системы в целом либо отдельных ее компонентов.

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

Словарь терминов

Access — доступ — 1) в обработке данных: взаимодействие между субъектом и объектом, обеспечивающее передачу информации между ними; 2) в физической безопасности: возможность входа на защищенную территорию.

Access control — управление доступом — в сетях или их компонентах: задачи, выполняемые аппаратурой, программным обеспечением и администрацией, с целью отслеживания выполняемых в системе операций, целостности данных, доступа и модификаций системных записей, выполнения пользователями идентификации и предоставление пользователям доступа.

Access period — период доступа — временной интервал, в течение которого действуют права доступа. В основном этот период определяется в днях или неделях.

Accountability — отслеживаемость — возможность для ответственных лиц восстанавливать ход нарушения или попытки нарушения безопасности информационной системы.

Accreditation — аккредитация — авторизация и санкционирование обработки критичных данных в операционной среде информационной системы или сети. Решение об аккредитации выносится после получения всеми лицами из технического персонала сертификата, подтверждающего возможность этих лиц работать с защищенными системами. При этом предварительно должно быть подтверждено соответствие проекта самой системы и его конкретной реализации набору заранее определенных технических требований. Все эти условия служат единственной цели — обеспечению степени безопасности адекватной уровню критичности данных.

Attack — атака — нарушение безопасности информационной системы, позволяющее захватчику управлять операционной средой.

Audit — контроль — образ действий, позволяющий получать независимый обзор и анализ системных записей и активности системы с целью установления ее текущего состояния безопасности.

Audit trail — след контроля — записи о транзакциях, выполняемых в системе, которые (записи) в совокупности документируют ход обработки информации в системе, что в свою очередь позволяет проследить (провести трассировку) его вперед — от исходных транзакций до создаваемых в процессе их работы записей и/или отчетов — а также назад — от конечных записей/отчетов до исходных транзакций. Последовательность записей, составляющих след контроля, позволяет определить источники возникновения транзакций в системе и последовательность их выполнения системой.

Auditing — ведение контроля — процедуры управления системой, необходимые для обеспечения нормальной работы системы и выполнения имеющихся задач, а также для обеспечения эффективности работы и эффективности использования ресурсов информационной системы. Ведение контроля может осуществляться лицами, отличными от лиц, непосредственно отвечающих за работу системы и решение конкретных задач.

Authentication — аутентификация — 1) процедура идентификации или проверки полномочности узлов, инициаторов или лиц для доступа к определенной категории информации; 2) план, разрабатываемый для обеспечения защиты от незаконных передач информации за счет установления санкционированности потоков информации, сообщений, узлов или инициаторов.

Authorization — авторизация — полномочия, устанавливаемые администратором системы для конкретных лиц, позволяющие последним использовать транзакции, процедуры или всю систему в целом.

Communications security — безопасность коммуникаций — аутентификация телекоммуникаций за счет принятия мер по предотвращению предоставления неавторизованным лицам критичной информации, которая может быть выдана системой в ответ на телекоммуникационный запрос.

Compromise — компрометация — утеря критичной информации, либо получение ее неавторизованными для этого субъектами (лицами, программами, процессами и т.д.)

Computer security — компьютерная безопасность — комплекс технологических и административных мер, применяемых в отношении аппаратных средств, программ, данных и служб с целью обеспечения доступности, целостности и конфиденциальности связанных с компьютерами ресурсов; сюда же относятся и процедуры проверки выполнения системой определенных функций в строгом соответствии с их запланированным порядком работы.

Confidentiality — конфиденциальность — некоторая классификация данных, получение либо использование которых неавторизованными для этого лицами может стать причиной серьезного ущерба для организации.

Cost-benefit analysis — анализ затрат/выгоды — стадия в разработке или развитии системы, на которой определяется стоимость обеспечения защиты данных в информационной системе; иногда под этой стоимостью подразумевают ущерб, который может быть нанесен в случае утери или компрометации данных, подлежащих защите.

Covert channel — скрытый канал — канал коммуникации, позволяющий процессу передавать информацию путем, нарушающим политику безопасности, реализуемую в данной системе.

Data integrity — целостность данных — состояние, при котором данные, представленные в компьютере, в точности соответствуют данным в исходных документах и при этом не могут быть подвержены неумышленным или умышленным искажениям или разрушениям.

Data protection — защита данных — средства защиты данных от

случайных или умышленных нежелательных модификаций, разрушений или разглашений.

Data security — безопасность данных — защита данных от неавторизованных, случайных, умышленных или возникших по халатности модификаций, разрушений или разглашений.

Data security officer (DSO) — офицер безопасности — лицо, отвечающее за обеспечение безопасности обработки данных в системе и за противодействие попыткам неразрешенного использования данных.

DOD Guidelines for Computer Security — совокупность правил, установленных Министерством Обороны США (DOD — Department of Defence) для определения степени безопасности узлового программного обеспечения — операционных систем, узловых средств контроля доступа и т.д.

Formal security policy model — формальная модель политики безопасности — политика безопасности, выраженная точным математическим образом, включающим начальное состояние системы, способы перехода системы из одного состояния в другое и определение "безопасного" состояния системы.

Hacking — неавторизованная попытка доступа к базе данных узла. Часто этот термин используется в отношении лиц, пытающихся получить доступ к узловой базе данных с удаленного узла за счет обхода (обмана) средств контроля доступа в сети.

Identification — идентификация — процесс анализа персональных, технических или организационных характеристик или кодов для получения (предоставления) доступа к компьютерным ресурсам.

Individual accountability — индивидуальный учет — комплекс мер, за счет которых идентификация пользователя может быть использована для определения возможности доступа пользователя к машинам, материалам и т.п.; правила предоставления пользователю времени, методов и режимов доступа.

Information security — безопасность информации — защита информационных массивов от случайных, либо преднамеренных, но неавторизованных разглашений, модификаций или разрушений, либо невозможность обработки этой информации.

Information system security — безопасность информационной системы — совокупность элементов, необходимых для обеспечения адекватной защиты компьютерной системы, включая аппаратные/программные функции, характеристики и средства; операционные и учетные процедуры, средства управления доступом на центральном компьютере, удаленных компьютерах и телекоммуникационных средствах; административные мероприятия, физические конструкции и устройства; управление персоналом и коммуникациями.

Integrity — целостность — состояние, в котором данные или программы используются установленным образом, обеспечивающим устойчивую работу системы; автоматическое восстановление в случае обнаружения системой потенциальной ошибки; автоматическое использование альтернативных компонентов вместо вышедших из строя. Примером является дублирование важных файлов с тем, чтобы в случае обнаружения ошибки или утери оригинального файла использовать его копию. Другим примером является поддержание двух и более путей доступа к устройству хранения.

Loophole — дыра, брешь — программное или аппаратное упущение или недоработка, позволяющая обойти процессы управления доступом. Синонимами являются термины fault, flaw.

Mandatory access control — полномочное управление доступом — ограничение доступа к определенным объектам, установленное на основании выявленной степени критичности информации, содержащейся в данном объекте. При этом обеспечивается "прозрачность" данного объекта для всех неавторизованных лиц, т.е. объект как бы невидим для них. Управление является полномочным, поскольку субъект с конкретными правами доступа не может передавать эти права другому субъекту.

Object — объект — сущность (т.е. запись, страница памяти, программа, принтер и т.д.), содержащая или получающая информацию. Если субъект имеет доступ к объекту, он (субъект) исходит из того, что объект предоставит ему доступ к хранимой им информации.

Object protection — защита объектов — средства защиты объектов типа сейфов, файлов и т.д. — того, что может быть выведено из защищенной области (protected area).

Object reuse — повторное использование объекта — предоставление некоторому субъекту доступа к магнитной среде, содержащей один и более объектов. Будучи доступной для субъекта, магнитная среда может в то же время содержать остатки данных от объекта, содержавшегося на этом месте ранее.

OMBA-123 — директива федерального правительства, определяющая, что при выполнении любыми правительственными органами или любыми компаниями работ для правительства с использованием компьютеров подрядчики должны составить краткий план идентификации и защиты ресурсов их информационной системы.

Operational data security — операционная безопасность данных — защита данных от модификации, разрушения или разглашения (случайных, неавторизованных, либо преднамеренных) во время выполнения операций ввода, обработки или вывода.

Orange book — "оранжевая книга" — полное название "Department of Defence Trusted Computer System Evaluation Criteria" DOD 5200.28-STD ("Критерий оценивания безопасности компьютерных систем министерства обороны") — государственный стандарт оценивания безопасности компьютерных систем, устанавливающий четыре иерархических класса — A, B, C и D — определенных уровней доверенности (иными словами, уверенности в безопасности) для конкретных приложений, разрабатываемых и используемых в интересах правительства.

3 Зак. 2503

3.1 Триединая задача защиты системы

Чтобы не допустить работы с системой какого-либо постороннего лица, случайно оказавшегося за терминалом, необходимо обеспечить распознавание системой каждого законного пользователя (или, по крайней мере, ограниченных групп пользователей). Для этого в некотором (желательно, защищенном) месте системы обычно заводится файл, куда записывается ряд признаков каждого пользователя, по которым можно этого пользователя опознать. В дальнейшем при входе в систему, а при необходимости — и при выполнении определенных действий пользователь обязан себя идентифицировать.

Идентификация пользователя заключается в том, что он при выполнении каких-либо действий должен себя назвать, вернее указать идентификатор, присвоенный данному пользователю в данной системе.

Получив идентификатор, система сравнивает его значение с эталоном, который хранится в указанном выше файле, и, в случае совпадения, обеспечивает возможность пользователю работать с компонентами системы.

Следует отметить, что идентификация не обязательно состоит во вводе некоторого имени с клавиатуры. Для идентификации пользователя могут применяться (и успешно применяются!) специальные устройства, способные идентифицировать пользователя по некоторым его физическим характеристикам, например по отпечаткам пальцев, спектральному составу голоса или даже по сетчатке глаза. Кроме того, для идентификации применяются различного рода магнитные карточки, специальные ключи и т.д.

После идентификации система обязательно производит аутентификацию полученного идентификатора: проверяется содержательность указанного идентификатора для данной системы. В самом деле, требование идентификации заключается только во вводе идентификатора. А если пользователь в ответ на запрос введет какое-нибудь случайное слово, первым пришедшее на ум? Если бы идентификация не дополнялась аутентификацией, то сама идентификация теряла бы всякий смысл. Часто аутентификация пользователя основана либо на запросе ввода пользователем пароля, либо на запросе ответов на некоторые тестовые вопросы. В любом случае для проведения аутентификации пользователь должен выполнить некоторые явные действия. Следует отметить, что механизмы, используемые для выполнения аутентификации, должны быть устойчивы к подлогу, подбору или подделке.

После распознавания пользователя современная система должна выяснить, какие права предоставлены этому пользователю: какие данные он может использовать (читать, писать, модифицировать, удалять); какие программы он может выполнять; когда, как долго и с каких терминалов он может работать (да, даже это!) и другие вопросы подобного рода. Выяснение системой всех этих вопросов называется "авторизацией" пользователя***.

Таким образом, выражение "пользователь авторизован для выполнения некоторых действий" означает только то, что пользователь имеет право (т.е. ему разрешено ответственными лицами компьютерной системы) выполнять в системе эти действия.

Авторизация обязательно проводится при входе пользователя в систему (т.е. в самом начале работы пользователя с системой). Собственно вход в систему практически полностью и состоит в выполнении идентификации, аутентификации и авторизации.

Однако в процессе последующей работы пользователя с системой обычно требуется возможность дополнительной авторизации пользователя в отношении работы с конкретными ресурсами или выполнения конкретных действий.

В настоящее время существуют самые разные механизмы реализации разделения доступа.

Одним из таких механизмов являются так называемые "списки управления доступом". Смысл их состоит в том, что каждому ресурсу системы при необходимости может быть сопоставлен некоторым образом организованный список, в котором указаны идентификаторы всех пользователей, которым разрешен (или, наоборот, запрещен) доступ к данному ресурсу, а также определено, какой именно доступ разрешен. При обращении некоторого пользователя к данному ресурсу система автоматически проверяет наличие у данного ресурса списка управления доступом и, если он есть, проверяет, разрешено ли данному пользователю работать с данным ресурсом в запрошенном режиме.

В качестве ресурсов могут выступать как отдельные объекты системы (файлы, устройства, носители, программы и т.д.), так и целые компоненты системы (все данные, все устройства или все программы) и даже вся система в целом.

Другим примером реализации механизма авторизации пользователя могут служить профили пользователей. Профиль пользователя — это тоже организованный некоторым образом список, сопоставленный определенному идентификатору пользователя и содержащий перечень всех объектов, к которым данному пользователю разрешен доступ, с указанием в каждом случае типа разрешенного доступа.

Существует также механизм, называемый "матрицей доступа". Матрица доступа — это некоторая системная структура данных, которую легче всего представить в виде таблицы, столбцы которой поме-

*** Здесь возможна некоторая путаница. Дело в том, что процесс проверки прав пользователя по доступу к системе называется авторизацией, но и вся совокупность прав пользователя также называется авторизацией. Более того, иногда под термином авторизация понимается вообще вся совокупность "идентификация/аутентификация/авторизация".

Конкретный смысл этого термина определяется обычно по контексту, в котором этот термин употреблен.

чены идентификаторами всех существующих в системе ресурсов, а строки — идентификаторами всех зарегистрированных в системе пользователей На пересечении каждого столбца таблицы с каждой ее строкой администратором проставляется специальный указатель разрешенного данному пользователю типа доступа к данному объекту.

Общим для перечисленных, а также для ряда других механизмов, обеспечивающих проведение системой в нужный момент авторизации пользователя, является следующее. Во-первых, доступ к данным механизмам должны иметь только специальные системные программы, обеспечивающие безопасность, а также строго ограниченный круг персонала системы, отвечающего за ее безопасность. Во-вторых, указанные механизмы сами должны быть тщательно защищены от случайного или преднамеренного доступа к ним лиц или программ, неавторизованных для этого, поскольку эти механизмы (данные и обслуживающие их специальные программы) являются одной из важнейших составляющих в обеспечении безопасности системы.

Многие известные атаки на системы нацелены на поражение или обход именно средств разделения доступа, поскольку они составляют одно из самых существенных препятствий на пути любого захватчика.

3.2 Общесистемные экспозиции

Помимо того, что сами системные программы могут быть нашпигованы самыми различными видами электронных ловушек, описанных выше, существует целый ряд методов проведения атак на безопасность системы в целом с целью получения доступа к отдельным ее компонентам. Приведем примеры таких методов:

1. BETWEEN LINES (между строк) — подключение к линиям связи и внедрение в компьютерную систему с использованием промежутков в действиях законного пользователя. Дело в том, что при интерактивной работе терминал большую часть времени простаивает: пока пользователь обдумывает что-либо, набирает команду, читает выведенную информацию и т.д. Эти “окна“ вполне могут быть использованы для работы с системой кого-либо под маской пользователя.

2. TRAFFIC ANALYSIS (анализ трафика) — захватчик анализирует частоту и методы контактов пользователей в системе. При этом вполне можно выяснить правила вступления в связь, после чего производится попытка вступить в контакт под видом законного пользователя.

3. LINE DISCONNECT (разрыв линии) — пользователь выходит из системы, либо захватчик разрывает линию, но система об этом не догадывается и продолжает работу с захватчиком как с законным пользователем.

4. MASQUERADE (маскарад) — захватчик использует для входа в систему ставшую ему известной идентификацию законного пользователя.

5. PIGGYBACK (свинство, или “подкладывание свиньи“) — захватчик подключается к линии связи и имитирует работу системы с целью получения информации об идентификации пользователя. Например, он может имитировать зависание системы и процедуру повторного входа в нее. Пользователь, не подозревая об этом, вводит свою идентификацию и пароль, после чего захватчик просто возвращает ему управление нормально работающей системой.

Особо следует остановиться на такой не слишком известной пользователям экспозиции системы, как “повторное использование объектов“.

Чтобы понять смысл этой опасности, следует вспомнить, что когда, например, операционная система сообщает вам о том, что по вашему желанию некоторый файл удален, то это вовсе не означает, что содержавшаяся в данном файле информация уничтожена в прямом смысле слова. Удаление файла означает только то, что система пометила блоки памяти, ранее составлявшие содержимое файла, специальным флажком, говорящим о том, что данный блок не входит в состав какого-либо файла и может быть поэтому использован для размещения в нем какой-то информации. Но та информация, которая была записана в данном блоке раньше, никуда не исчезает! То есть, если прочитать сам блок (а это вполне можно сделать), то можно беспрепятственно получить доступ к “удаленной“ информации!

Описанная стратегия “удаления“ по ряду вполне объективных причин используется в подавляющем большинстве компьютерных систем. Причем в качестве объекта атаки могут фигурировать не только блоки файлов, но и различного рода буферы, кадры страниц памяти, секторы магнитных дисков, зоны магнитных лент, регистры памяти и т.д.

Вполне логичным представляется то, что описанная выше стратегия удаления (а вернее всего — сохранения, как это ни парадоксально звучит!) провоцирует различного рода захватчиков на попытки считывания остатков информации прямо из памяти машины. Для этого в ряде случаев бывает достаточно создать небольшую программку, запрашивающую во время выполнения динамического выделения дополнительной памяти достаточно большого объема. Затем в результате умышленной ошибки эта программа может аварийно завершаться с выдачей так называемого “посмертного“ дампа, представляющего собой длинную распечатку в шестнадцатиричном представлении содержимого всех областей памяти, используемых “свалившейся“ программой. Читать такую распечатку, тем более искать в ней что-либо без специальных навыков сложно, поэтому рядовой оператор может и не понять, что буквально на его глазах осуществляется кража информации.

Весь смысл такой программки состоит в том, чтобы захватить в свое распоряжение как можно больше областей памяти, только что использованных программой, обрабатывавшей критичную информацию, в надежде извлечь из них остатки этой критичной информации. Конечно, вероятность выделения программе-захватчику именно тех областей, которые со-

держат остатки интересующей захватчика информации, достаточно мала. Однако такая вероятность существует, особенно если программу-захватчика запустить сразу же по завершении работы программы, обрабатывавшей критичную информацию.

В конце концов, получить дамп интересующих областей памяти можно и с использованием ряда стандартных системных утилит — и это лишний раз доказывает необходимость ограничения доступа пользователей даже к обычным средствам системы.

Борьба против атак такого рода заключается в устранении возможности считать остатки информации. Делается это либо полным затиранием остатков информации или заполнением их какой-либо бессмыслицей (в простейшем случае — просто блоком сплошных нулей или единиц), либо — более тонко — путем отказа любому пользователю в возможности прочитать "свободный" блок до тех пор, пока этот пользователь полностью не заполнил его своей собственной информацией.

В настоящее время ряд операционных систем изначально содержат встроенные средства блокировки "повторного использования". Для других типов операционных систем существует достаточно много коммерческих программ, не говоря уже о специальных пакетах безопасности, реализующих аналогичные функции.

3.3 Четыре базовых средства обеспечения безопасности

Существуют четыре универсальных метода защиты и противодействия нарушениям безопасности компьютерных систем.

Первый метод — шифрование данных. Шифрованием называется некоторое обратимое однозначное преобразование данных, делающее их непонятными для неавторизованных лиц. Шифрование насчитывает тысячелетнюю историю, во всяком случае имеются сведения, что к шифрованию донесений прибегали еще древнегреческие полководцы. Чтобы получить наглядный пример шифрования, советую вам прочитать рассказ Конан Дойля "Пляшущие человечки" о Шерлоке Холмсе или роман Жюля Верна "Граф Матиас Шандор". Помимо названных произведений существует также масса специальной литературы по данному вопросу.

Специалисты считают, что шифрование является одним из самых надежных средств обеспечения безопасности данных. В самом деле, охота за данными в компьютерных системах во многом обусловлена тем, что слишком многие люди, не имеющие сколь-нибудь специального образования, способны разобраться, как использовать эти данные в своих личных интересах. Но(!) все это имеет смысл лишь при условии, что сами данные представлены в понятном виде, например в виде строки текста на некотором естественном языке. Однако стоит произвести несложную манипуляцию — провести однозначную замену одних символов (допустим, букв) на другие символы (допустим, цифры в сочетании со специальными символами) как чтение той же строки текста будет весьма серьезно затруднено.

Метод защиты информации шифрованием подразумевает обязательное выполнение следующих требований. Никто, кроме хозяина данных и лиц, которым разрешен доступ к этим данным, не должен знать, во-первых, самого алгоритма преобразования данных, а, во-вторых, управляющих данных для такого алгоритма — так называемых ключей.

Доступ к зашифрованным данным обязательно включает этап расшифровки данных — т.е. выполнения обратного преобразования данных из непонятного в понятное представление.

Шифрование делает почти бессмысленным простой доступ к данным: ведь, не зная ключа, захватчик может годами биться над украденной абракадаброй, но так и не понять смысла данных. Кроме того, шифрование имеет еще одно немаловажное в смысле безопасности свойство. Расшифровка возможна только в том случае, когда зашифрованные данные не были искажены. Искажение зашифрованных данных в силу однозначности преобразования неминуемо повлечет искажение данных, получаемых в результате расшифровки. Таким образом, если захватчик как-либо исказил (модифицировал) зашифрованные данные, то факт нарушения безопасности будет выявлен при первой же попытке расшифровки, поскольку в расшифрованных данных появятся искаженные участки.

Это свойство особенно полезно для предотвращения искажений программ. Код программы шифруется с тем, чтобы быть расшифрованным только на время выполнения программы. Если в промежутке между выполнениями программы ее зашифрованный код исказить, то после расшифровки программа в большинстве случаев работать не будет. Хотя и есть некоторая вероятность случайного искажения зашифрованного кода таким образом, что программа все же будет работать, однако вероятность эта очень мала. Еще меньше вероятность успешного выполнения целенаправленного изменения зашифрованного кода программы, например, с целью вставить в нее одну из ловушек.

В идеале данные в расшифрованном виде должны существовать только во время их законной обработки владельцем и быть зашифрованными во всех других случаях — при хранении, пересылке по линиям связи и т.д. Однако такое тотальное шифрование существенно снижает эффективность системы, так как ее значительные ресурсы отвлекаются на выполнение операций шифрования/расшифрования. Приемлемый вариант обычно достигается путем определения для такого способа защиты ограниченного количества данных, требующих наиболее высокого уровня защиты.

Второе универсальное средство защиты — регулярное создание резервных копий системы целиком или ее наиболее важных компонентов.

Имея в запасе резервную копию и столкнувшись с искажением ваших данных или их потерей, вы просто извлекаете копию утерянных данных и продолжаете нормальную работу. Даже если с момента создания резервной копии до момента утери данных прошло некоторое время, в течение которого вы вносили в эти данные какие-либо изменения, наличие резервной копии значительно облегчит вам восстановление данных, поскольку проще восстановить внесенные изменения, чем заново подготовить все данные целиком.

Полезность резервного копирования и его важность трудно переоценить. Наличие резервных копий (лучше всего — нескольких, хранимых в разных местах) обезопасит вас от многих неприятностей. В частности, наличие резервной копии системы — уникальный по универсальности способ борьбы с компьютерными вирусами. Многими центрами принято хранить три последних копии системы по схеме "дед-отец-сын". Кроме того, важным считается установление определенной дисциплины и порядка копирования. Нарушение этой дисциплины карается не менее серьезно, чем любое нарушение безопасности.

Конечно, создание и хранение резервных копий требует определенных затрат ресурсов системы, равно как и значительного терпения ответственных за копирование лиц. Но, во-первых, все затраты с лихвой окупаются за счет обеспечения безопасности системы, а, во-вторых, в современных системах существует немало средств, специально предназначенных для создания резервных копий и существенно облегчающих выполнение копирования.

Третье средство предупреждения неприятностей — наличие так называемого следа контроля. Для любого захватчика серьезным поводом подумать о целесообразности атаки служит факт наличия в компьютерной системе средств осуществления контроля. Контроль системы заключается в выделении, накоплении в едином месте (так называемом "следе контроля"), защищенном хранении и предоставлении по требованию авторизованного (для выдачи такого требования) пользователя специальных данных о различных типах событий, происходящих в системе и так или иначе влияющих на состояние безопасности данной компьютерной системы.

След контроля не тождествен контрольному журналу; контрольный журнал — всего лишь частный вид реализации следа контроля. По определению след контроля — это совокупность записей, содержащих требуемую контрольную информацию, представленную в установленном формате. Таким образом, след контроля может быть реализован в виде специальной области памяти, доступ к которой может быть осуществлен исключительно средствами подсистемы контроля, либо в виде отдельного устройства, опять же защищенного от незаконного доступа со стороны неавторизованных для этого пользователей.

Контроль системы преследует в основном две цели. Он служит:

a) для отслеживания текущего состояния безопасности в защищаемой системе, своевременного обнаружения возможности нарушения безопасности и предупреждения об этом лиц, отвечающих за безопасность системы;

b) для обеспечения возможности обратной трассировки (по данным контроля) происшедшего нарушения безопасности с целью обнаружения причин данного нарушения и установления степени ответственности причастных к нарушению лиц.

Четвертое средство — использование избыточных данных. В некотором смысле резервная копия — частный случай избыточных данных. Но не единственный.

Например, часто практикуется хранение в некотором защищенном месте системы так называемых сигнатур важных объектов системы.

Например, для файла в качестве сигнатуры может быть использовано сочетание байта защиты файла с его именем, длиной и датой последней модификации. При каждом обращении к файлу или в случае возникновения подозрений текущие характеристики файла сравниваются с эталонными. Расхождение каких-то характеристик свидетельствует о возможной модификации файла (например, может увеличиться или уменьшиться длина файла).

Другим примером избыточных данных является применение контрольных сумм, контроль данных на чет-нечет, помехоустойчивое кодирование и т.д. В любом случае применение избыточных данных направлено на предотвращение появления в данных случайных ошибок и выявление неавторизованных модификаций.

И. Моисеенков

Использованы материалы:

D. Tassel, "Computer Security Management", Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1972.

D. Elizabeth, R. Denning, "Cryptography and Data Security", Purdue University, AddisonWesley Publishing Company, 1982.

D. Davies, W. Price, "Security Computer Networks", John Wiley & Sons, 1984.

C. Pfleeger, "Security in Computing", 1988, University of Tennessee and Trusted Information Systems, Inc.

E. Spafford, "The Internet Worm Programm: An Analysis", ACM Commitee Report, 1989.

Datapro Report, January 1989.

P. Fites, P. Johnston, M. Kratz, "The Computer Virus Crisis", 1989.

"Computer & Security", 8(1989).

"КомпьютерПресс", №1-2 (1989), №3-10 (1990).

(Окончание следует)



Вероятно, трудно найти в нашей стране пользователей персонального компьютера, которому совершенно не известно, что такое PC Tools, хотя этот популярный программный продукт рассчитан на более или менее подготовленного пользователя, уже чувствующего себя профессионалом.

PC Tools 7.0 – интегрированный профессиональный инструментарий

Изготовитель пакета PC Tools, американская фирма Central Point Software, входит в число мировых технологических лидеров по производству программ. Эта фирма только что официально объявила о завершении работ над новейшей версией пакета — PC Tools 7.0. Central Point Software утверждает, что этот новый пакет утилит пока является единственным на рынке инструментальным средством, идеально подходящим для всех пользователей персональных компьютеров, которые используют в своих машинах новую версию DOS 5.0 или Windows 3.0.

Благодаря безусловному успеху предыдущей версии PC Tools 6.0 не только окрепла всемирная слава Central Point Software как фирмы, следующей в ногу с прогрессом в информационных высоких технологиях, но и существенно возросли прибыли от реализации программных средств. По известным данным, опубликованным в прессе, доходы фирмы Central Point Software увеличиваются по меньшей мере на 50% ежегодно. Всем известно, что на Западе рынок любых продуктов информационной технологии в очень большой степени зависит от паблисити, и поэтому он тесно связан с отзывами и оценками на страницах ведущих компьютерных журналов. Тем самым многочисленные компьютерные журналы давно уже превратились в органическую часть рынка программ и всевозможного компьютерного “железа“, ибо страницы журналов — это и есть грандиозная витрина этого рынка. Если новый продукт не будет положительно отмечен на страницах компьютерных журналов, рассчитывать на успешный маркетинг и продвижение этого товара на рынок практически невозможно.

В 1990 году пакет PC Tools 6.0 получил высочайшую оценку экспертов редакции журнала PC Magazine, почетный приз журнала InfoWorld за лучший продукт 1990 года под названием “Product of The Year“, приз журнала PC World “1990 Word Class Award“ и множество других призов, наград и восторженных рецензий, продолжающих появляться на страницах почти всех компьютерных периодических изданий мира. Например, журнал BYTE только что сообщил в июльском номере, что пакет PC Tools Deluxe 6.0 завоевал приз читательских симпатий “1991 Reader´s Choice Award“.

Расположенный в Мюнхене филиал фирмы Central Point Software Deutschland GmbH любезно предоставил нам для ознакомления бета-копию новейшей версии PC Tools 7.0, чтобы заблаговременно оценить ее и проинформировать наших читателей о возможностях этой "интегрированной группы утилит", как называют этот пакет сами изготовители. Исследование бета-версии PC Tools 7.0 осуществили сотрудники малого предприятия "Такт". Знакомство с новой бета-версией любой программы, словно генеральная репетиция спектакля, обычно не предназначена для публичного обсуждения обнаруженных недостатков. Поэтому здесь вы сможете узнать только лишь одни хорошие новости, характеризующие лучшие стороны нового программного продукта, а критические замечания о PC Tools 7.0 мы сможем высказать позже, проанализировав уже готовое изделие.

Новая версия PC Tools 7.0 имеет целый ряд последовательных усовершенствований по сравнению с PC Tools 6.0. Помимо дюжины новых функций и свыше сотни улучшений в работе прежних функций, в этой версии предусмотрена возможность работы как под управлением DOS, так и в графической среде Windows 3.0. Выполнение многих функций PC Tools 7.0 превосходно дополняет новую операционную систему DOS 5.0, так как создатели пакета стремились к полнейшей программной совместимости и комплиментарности с DOS 5.0. PC Tools 7.0 автоматически поддерживает новую возможность DOS 5.0 размещаться в "верхней памяти", освобождая тем самым большую часть основной оперативной памяти для размещения прикладных программ. С утилитами PC Tools 7.0 совместима и новая оболочка DOS 5.0 Shell, используя которую вместо оболочки PCShell, входящей в PC Tools 7.0, можно успешно применять многие утилиты PC Tools 7.0 для восстановления, резервного копирования, поиска и просмотра файлов разных форматов.

Кроме того, PC Tools 7.0 имеет развитую поддержку для работы в локальной сети под Novell NetWare. При работе PC Tools 7.0 под DOS на экране возникает весьма удобный и визуально очень похожий на Windows графический интерфейс пользователя, соответствующий получающему все более широкое распространение стандарту IBM CUA (Common User Access). Существенно облегчает работу с PC Tools 7.0 и мощная система контекстной гипертекстовой справки. Быстро и легко перемещаться внутри этой разветвленной help-системы позволяет гипертекст: указав мышью выделенные слова или понятия и лишь щелкнув кнопкой, вы мгновенно переходите к более глубокой и подробной информации следующего уровня. Все команды в PC Tools 7.0 можно вводить и в командной строке, но едва ли пользователи станут пренебрегать эффектной и удобной системой меню. Специальная система изменения конфигурации и расцветки меню показалась нам даже чрезмерно богатой альтернативами. Можно незаметно для себя потратить уйму времени, подбирая и экспериментируя со всевозможными вариантами оттенков меню, надписей и опций. Слишком обширный выбор может поставить в тупик пользователя, воспитанного в среде нашего извечного дефицита!

Пожалуй, едва ли можно назвать "хорошей новостью" то, что новая версия PC Tools 7.0 оказалась, увы, еще менее компактной, чем предыдущая. Но это можно уже признать вполне естественной тенденцией в развитии современного программного обеспечения, которое явно не желает равняться на бедные техническими возможностями персональные компьютеры, каких в нашей стране подавляющее большинство. Полученная нами бета-копия PC Tools 7.0 занимает 8 дискет по 720 Кбайт, причем часть программ на этих дисках помещена в сжатом виде. При инсталляции полного набора утилит PC Tools 7.0 для использования под DOS требуется 5,9 Мбайт свободного пространства на жестком диске. Если вы собираетесь использовать ее в качестве приложения для Windows 3.0, потребуется только 1,9 Мбайт дискового пространства, помимо около 5 Мбайт, занимаемых Windows. Кроме того, при работе под DOS необходим минимум 640 Кбайт памяти, а для использования PC Tools 7.0 вместе с Microsoft Windows 3.0 нужно иметь не менее 2 Мбайт оперативной памяти. Такие технические требования могут кому-то показаться чрезмерно жесткими и отсекающими часть рынка более "слабых" персональных компьютеров. Но воспринимать это можно и как заявку на рынки сбыта в будущем, когда через пару лет возможности средних стандартных персональных компьютеров наверняка станут обгонять нынешние высшие достижения.

Приводимый ниже краткий обзор возможностей многочисленных утилит, составляющих новую PC Tools 7.0, показывает, что многие из них существенно не изменились в сравнении с предыдущей версией. Это вполне соответствует известной концепции маркетинга программных продуктов, которая гласит: "торопись медленно", то есть в каждой последующей версии программы должна сохраняться логическая преемственность с предыдущими версиями. Несомненно, эта заповедь в PC Tools 7.0 свято соблюдена, и те пользователи, которым уже знакомы предыдущие версии, смогут без труда адаптироваться к переменам и новшествам новой версии.

Резервное копирование жесткого диска в DOS и Windows

Утилита Backup включает полную совместимость данных и полную идентичность манипуляций клавишами при использовании приложений под DOS и Windows. Новый очень выразительный интерфейс пользователя внешне очень похож как для DOS, так и для Windows.

Резервное копирование может осуществляться в фоновом режиме в процессе работы в приложениях Windows. Возможно использование автоматического режима выполнения резервного копирования по установленному графику. Поддерживается широкий перечень

различных устройств для резервного копирования на гибкие и жесткие диски, на кассеты Бернулли и на стримеры формата QIC-40/80 и Irwin. Допустимо резервное копирование записей длиной до 250 Мбайт на кассету с магнитной лентой. Скорость передачи данных — до 7 Мбайт/мин при записи на магнитную ленту и до 3,5 Мбайт/мин при записи на гибкие диски или другие устройства DOS. При этом данные могут сжиматься до 60% для экономии места на носителях со снижением скорости записи.

В целях повышения надежности перед записью осуществляется верификация носителя резервной копии. Возможно побитовое сравнение для выбора текущих файлов для копирования. Также возможна коррекция ошибок при извлечении данных с поврежденных дискет, жестких дисков и магнитных лент.

Восстановление утраченных данных

Эта традиционная для PC Tools функция предназначена для восстановления данных на диске, утраченных вами почти по любым возможным "сценариям": после уничтожения или повреждения файлов, из-за нарушений в работе жестких дисков и даже из-за нечаянного форматирования дисков. Четыре различные утилиты выполняют эту важную функцию.

Утилита Diskfix консультирует пользователя о состоянии и использовании дисков, объясняет, как следует понимать сообщения DOS об ошибках, производит автоматическую диагностику гибких и жестких дисков, а также их последующее "лечение". Причем команда "Undo" все изменения позволяет опять привести в прежнее состояние. Могут подвергаться "лечению" любые диски с нарушенной таблицей разбиения (Partition) диска, с поврежденной загрузочным (Boot) сектором, с потерянными кластерами, с поврежденной таблицей расположения файлов (FAT), разрушенным корневым каталогом и подкаталогами или с файлами, которые "переплелись" друг с другом.

Утилита FileFix позволяет восстанавливать поврежденные файлы форматов dBASE, Lotus 1-2-3 (в том числе и версии 3.x) и Symfony.

Превосходная утилита Undelete восстанавливает уничтоженные файлы как в отдельном компьютере, так и в локальной сети. Эта утилита также работает под DOS и Windows. При использовании для уничтожения файлов утилиты Sentry, не позволяющей использовать для перезаписи освободившееся пространство на диске, практически всегда возможно полное восстановление уничтоженных данных, даже в сети.

Уничтоженные файлы сортируются по дате, времени и состоянию. Даже если уничтожен подкаталог, возможно восстановление файлов, баз данных и электронных таблиц. Кроме того, возможно детальное восстановление сильно поврежденных файлов вручную. С помощью этой утилиты возможно восстановление файлов, разорванных на отдельные фрагменты, например, после неудачной "оптимизации" диска, после переписывания или перемещения имен каталогов, частично — после перезаписи файлов или после случайного форматирования диска.

Утилита Unformat "ремонтирует" жесткий диск: вновь воссоздает утраченную таблицу FAT по ее "отраженному" образу в специальном защищенном файле Mirror, который всегда находясь в корневом каталоге на диске, сохраняет и регулярно обновляет информацию, необходимую для восстановления таблицы расположения файлов. При этом после "деформатирования" диска гарантируется 100%-ное восстановление утраченных данных, если диск был отформатирован специальной утилитой PC Tools 7.0.

Защита данных и дисков

С помощью нескольких утилит ваши ценные данные могут быть надежно защищены от потерь из-за случайного форматирования диска, заражения вирусами, нарушений в работе носителей или потери памяти CMOS.

В PC Tools 7.0 имеется весьма полезная утилита для безопасного освежения адресов дисковых секторов, которые записываются только однажды при форматировании и обычно со временем стареют и размагничиваются. Рано или поздно это приводит к потере данных или даже к полному "крушению" жесткого диска, когда вдруг ваш компьютер начинает выдавать неприятные сообщения типа "Sector not found" или "Seek error". Эта утилита PC Tools 7.0 тщательно сканирует всю поверхность диска, производя испытания на чтение/запись по 80 параметрам, чтобы обнаружить и отбраковать вновь возникающие ненадежные секторы. Затем корректируется фрагментация файлов на диске и настраивается фактор чередования сегментов для оптимизации скорости работы жесткого диска.

Специальная резидентная программа, занимающая 20 Кбайт в ОЗУ, способна, как утверждает фирма Central Poin Software, распознать свыше 600 вирусов в любом файле, каталоге, в памяти или в загрузочном секторе. При диагностировании вируса после его поступления в компьютер по сети или с дискетами, она немедленно отключает жесткий диск, дает сообщения новых версий вирусов. Служба поддержки Central Point Software обязуется регулярно снабжать зарегистрированных пользователей обновленными версиями этой утилиты.

Таблица расположения файлов на диске (FAT), загрузочный (Boot) сектор и корневой каталог любого диска могут сохраняться в специальном защищенном файле, создаваемом утилитой Mirror, на тот случай, если понадобится быстро и надежно восстановить диск после неожиданного аварийного отключения питания или после случайного форматирования диска.

Специальная загрузочная дискета, создаваемая PC Tools 7.0, позволяет перезагрузить компьютер после любой неисправности, так как на ней имеются все необходимые для восстановления файлы, включая загру-

4 Зак. 2503

зочный сектор, таблицу разбиения диска и образ CMOS-памяти.

Сервисная система Desktop

С помощью системы Desktop можно легко и удобно организовать групповое использование персонального компьютера, вести всевозможные записи в записной книжке и в простой базе данных, пользоваться календарем и калькулятором, составлять списки "что сделать", а также получать в установленный срок напоминания, скажем, о назначенной встрече или о совещании с коллегами в локальной сети. Эта же система позволяет производить регулярное резервное копирование данных, автоматически отправлять в удобное время факсимильные сообщения и электронную почту.

База данных системы Desktop совместима с файлами формата dBASE, что позволяет легко пользоваться имеющимися файлами, создавать и редактировать новые файлы, составлять списки и отчеты, "сливать" тексты стандартных писем с адресами и осуществлять автоматический набор телефонных номеров через модем.

В простом текстовом процессоре, встроенном в Desktop, можно легко создавать любые по объему документы. В нем имеется функция Mail-merge, позволяющая "сливать" тексты писем с адресами из базы данных. На экран могут выводиться одновременно до 15 "листков" из электронной записной книжки. В текстовом процессоре имеются практически все важнейшие функции специализированных программ такого типа: автоматическая проверка правописания английского языка, поиск и замена фрагментов текста, перемещение фрагментов текста, форматирование страницы перед печатью.

В PC Tools 7.0 имеется сразу 4 встроенных калькулятора — обычный алгебраический и специальные для научных и финансовых расчетов, а также для программистов. Автоматически через модем могут набираться номера любых телефонов, либо вводимых в командной строке, либо сохраняемых в системе Desktop или в любой другой прикладной программе. Трудно кратко перечислить все возможности, представляемые в системе Desktop. Вот для примера еще несколько.

С помощью утилиты Clipboard в системе Desktop можно "вырезать" любые данные, отображаемые на экране, и вставлять их в различные приложения. Утилита Marco editor позволяет системе запоминать любые длинные комбинации клавиш, которыми вам приходится часто пользоваться, а затем воспроизводить всю эту макрокоманду после нажатия лишь одной условной клавиши. А чрезвычайно полезная утилита Outliner предназначена для составления планов и расписаний, оглавлений и списков важных мероприятий, встреч, покупок и тому подобных дел и событий, из которых состоит жизнь любого занятого человека, для разработки и обдумывания структуры различных документов, докладов, речей, романов.

Управление дисками и файлами

Утилиты FileFind и View позволяют весьма удобно и оперативно находить по любым признакам (а в текстовых файлах даже по образцу фрагмента текста!) и просматривать файлы данных различного формата из более чем 35 распространенных прикладных программ: из практически всех наиболее популярных текстовых процессоров, электронных таблиц, систем управления базами данных, а также просматривать графические файлы формата PCX из программы PC Paintbrush и из архивов форматов ARC, LZH, PAK, PKZIP и ZOO.

Оболочка DOS, предлагаемая в PC Tools 7.0, на наш взгляд, более удобна и гораздо богаче возможностями, чем, скажем, известная оболочка Norton Commander или DOS Shell версии 4.01. Впрочем, поскольку "на вкус и цвет товарищей нет", выбор подходящей оболочки остается делом весьма индивидуальным: последнее слово в окончательном выборе всегда должно оставаться за пользователем. Достаточно назвать несколько отличительных особенностей имеющейся оболочки DOS: просмотр содержимого двух файлов одновременно, копирование файлов на несколько дисков, возможность резидентного расположения в памяти (оболочка занимает при этом всего 10 Кбайт), переназначение функциональных клавиш по вашему вкусу и многое другое.

Помимо перечисленных возможностей, оболочка PC Tools 7.0 позволяет защищать имеющиеся прикладные программы от нежелательного доступа посторонних лиц секретным паролем. Замечательно, на наш взгляд, выглядит и графическое представление объема каталогов (точнее, того, сколько места они занимают на диске) при просмотре дерева каталогов на экране.

Системная и сетевая информация

PC Tools 7.0 производит всестороннее тестирование системы утилитой SI и сообщает пользователю более чем о 160 параметрах, исчерпывающе характеризующих используемую компьютерную систему и подключенную коммуникационную сеть.

Помимо многочисленных полезных данных о типе используемого процессора и его производительности, о версии операционной системы, о прерываниях, о типе видеоадаптера, о различных текстовых параметрах скорости чтения/записи дисководов и другой внутренней статистике, на экране графически отображается информация о распределении памяти. Детально сообщается об используемых физических и логических дисках (включая сетевые) и о виртуальных RAM-дисках.

Утилита SI подробно сообщает также о всех сетевых параметрах, о версии и изготовителе сетевого программного обеспечения, дает детальный и обобщенный отчет о каждом сетевом сервере, составляет списки серверов, список пользователей по каждому серверу, а также выводит всевозможную

информацию о группах пользователей и о пространстве сервера для каждого клиента.

Графически в виде гистограмм отображаются результаты теста Benchmark — сравнительная скорость работы процессора и дисков, обобщенная производительность системы, а также график производительности сети по времени чтения/записи на сервер. Кроме того, не выходя из PC Tools, можно просматривать файлы AUTOEXEC.BAT и CONFIG.SYS, а также получить другие данные о конфигурации системы.

Использование компьютера на расстоянии

Система связи Commute пакета PC Tools 7.0 позволяет с помощью, например, домашнего персонального компьютера или переносного laptop-компьютера запускать прикладные программы, работающие под управлением DOS и Windows в компьютерах или в локальной сети где-то на удалении, скажем, на вашем предприятии или в учреждении. Передача информации по кабелю, по телефонной линии через модем или в локальной сети ускоряется вдвое, благодаря компрессии данных. Можно распечатывать файлы, управлять приложениями Windows и пользоваться мышью либо на собственном, либо на удаленном компьютере. Система Commute функционально совместима с сетями типа Novell (версия 2.15 и выше) и позволяет получать и передавать электронную почту и получить доступ к любому персональному компьютеру сети через модем.

Ваш компьютер может быть защищен от нежелательного доступа секретным паролем, автоматически отзываясь и требуя подтверждения права доступа. Причем защита от несанкционированного доступа может быть многоуровневой, а вся запрашиваемая информация регистрируется. Эта система Commute имеет множество других дополнительных полезных сервисных удобств, предназначенных для существенного облегчения работы в локальной сети.

Оптимизация технических характеристик системы

Помимо уже упоминавшихся утилит, позволяющих диагностировать и максимально оптимизировать скорость работы жесткого диска, благодаря дефрагментации и реорганизации расположения файлов на диске, а также благодаря оптимальной настройке фактора чередования в последовательности расположения секторов в каждом из цилиндров жесткого диска, в PC Tools 7.0 имеется дополнительная возможность ускорить операции чтения/записи жесткого диска, за счет сохранения наиболее часто используемых данных в кэш-памяти.

Утилита Disk Cache также работает под Windows и позволяет использовать расширенную и дополнительную (EMS) память компьютера. В кэш-памяти одновременно могут сохраняться данные, относящиеся сразу к четырем жестким дискам.

Телекоммуникация

С помощью встроенных утилит коммуникации, входящих в PC Tools 7.0, ваш компьютер может быть присоединен к другим компьютерам как внутри вашего предприятия или учреждения, так и к любому персональному компьютеру в любом конце света, используя связь по телефонным линиям через модем, факс-плату или подключение к службам электронной почты типа MCI mail, CompuServe или Easy-Link.

Процесс передачи и получения электронной почты максимально автоматизирован. Поддерживается эмуляция терминалов ANSI, VT100 и VT52. Через модем можно передавать файлы, используя протоколы XMODEM или ASII. Факсимильные сообщения можно посылать и получать как на обычный аппарат факс, так и на другой персональный компьютер, снабженный встроенной факс-платой в разъеме расширения.

Безопасность хранения данных

Учитывая ценность информации, которая может храниться в персональных компьютерах, богатый выбор разнообразных мер обеспечения безопасности и секретности, предлагаемых пакетом PC Tools 7.0, пожалуй, никак нельзя считать излишним.

Любая конфиденциальная информация, содержащаяся в ваших файлах и каталогах, находящихся на дисках и передаваемых в сетях, может включать автоматическую шифровку по вашему собственному секретному паролю.

Кроме того, предусмотрена защита указанных пользователем файлов, каталогов и системных областей или всего диска от случайной перезаписи. Экран монитора, содержащий конфиденциальную информацию, может не только очищаться (и тем самым продлевать срок службы монитора), но и имеет встроенную защиту по паролю, чтобы в ваше отсутствие ограничить нежелательный доступ к компьютеру для посторонних лиц. И, наконец, имеется традиционная для PC Tools утилита Wipe, позволяющая бесследно уничтожать ненужные данные в файлах или на всем диске.

Минимальные требования к системе

Пакет PC Tools 7.0 предназначен для работы на персональных компьютерах типа IBM PC, XT, AT, PS/2 или на 100%-совместимых машинах, имеющих 640 Кбайт ОЗУ и DOS 3.0 или выше. Применение DOS 5.0 рекомендуется. Приложения Windows могут выполняться, если имеется Microsoft Windows 3.0 или выше и 2 Мбайт ОЗУ. В любом случае необходим жесткий диск. Поддерживаются драйверы мыши Microsoft Mouse версии 6.14 или выше, а также драйверы мыши Logitech/Dexxa версии 3.4 x или выше либо 100%-совместимые. Предполагается использование модема типа Hayes, поддерживается NetWare 2.15 или выше и NetWate 386.

Хотя бета-версии программ обычно не вполне совпадают с готовым продуктом (в конце концов, для того они и создаются, чтобы искать в них возможные ошибки и недостатки), без риска ошибиться можно уже сегодня предположить, что новому пакету PC Tools 7.0 уготована не менее успешная судьба, чем его предшественнику. Преемственность испытанных и уже нашедших широкое признание инструментов в PC Tools 7.0 не только тщательно сохранена, но и существенно обогащена новыми мощными и полезными возможностями, хотя в этой статье невозможно было упомянуть и прокомментировать каждую из них. Но даже некоторые из перечисленных возможностей PC Tools 7.0 могут сегодня вызвать зависть и огорчения у некоторых читателей или показаться явно излишними из-за скудных технических возможностей отечественного парка компьютеров и сетей коммуникации. Кто-то может даже сказать, что "они там с жиру бесятся" или пробурчать что-то об "элитарности" утилит для Windows.

Не стоит, однако, забывать, что создается этот пакет отнюдь не по нашей мерке, а прежде всего для западного рынка. Во-вторых, прогресс все-таки происходит и у нас, хотя, быть может, и не столь стремительно, как нам того хотелось бы, а каждый шаг в улучшении проклятого "железа" заставляет раскошеливаться. Однако рано или поздно большинство этих возможностей PC Tools 7.0 станут полезными и актуальными для многих. Ну и, наконец, в-третьих, процесс инсталляции позволяет выбирать из обширного интегрального инструментария PC Tools 7.0 только те инструменты, которые вам нужны или могут быть полезны именно сегодня, и тем самым позволяет сэкономить используемое дисковое пространство и свободную память. Пользователь может даже "дезинтегрировать" эту систему и выбрать из нее лишь несколько самых необходимых утилит, которыми можно пользоваться даже на самом скромном персональном компьютере.

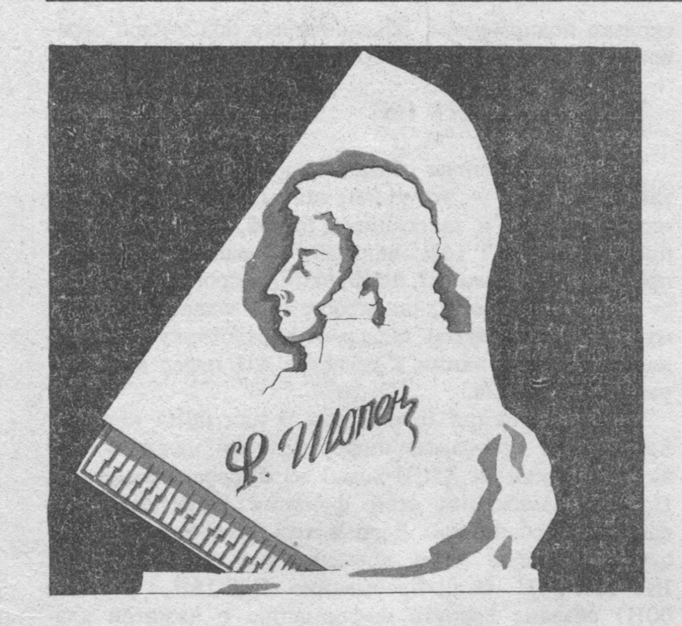
Девиз Central Point Software гласит; "Сделаем так, чтобы работа на компьютере стала безопаснее, проще, быстрее". Несомненно, представленный сегодня интегрированный пакет утилит PC Tools 7.0 — это верный очередной шаг в указанном стратегическом направлении. Ну, а каков окажется окончательный вариант этой программы, надеюсь, все мы скоро сможем узнать.

Для желающих связаться с мюнхенским филиалом Central Point Software, который отвечает за маркетинг и поддержку продуктов этой фирмы в ФРГ, Австрии, Швейцарии и в странах Восточной Европы, сообщаем контактные реквизиты:

Central Point Software Deutschland GmbH
Leopoldstr. 28 a/II
8000 Munchen 40
BRD

Телефон: (089) 33 67 17
Факс: (089) 33 57 31

А.В. Петроченков
Телефон: (08100) 5 58 05



С момента нажатия клавиши на клавиатуре до отображения прикладной программой символа на экране совершается целая последовательность событий. Эта статья начинается с обзора прерываний, обслуживающих события, связанные с клавиатурой. Затем рассматриваются программные и аппаратные средства, относящиеся к клавиатуре и ее прерываниям; непонятная техническая терминология попутно поясняется. Далее предлагаются примеры программ, демонстрирующих внутреннюю работу клавиатуры. И, наконец, представлены некоторые утилиты, улучшающие функциональные возможности клавиатуры. Статья перепечатывается из журнала Microsoft System Journal, v.5, No.1, 1990 с любезного разрешения фирмы Microsoft.

Клавиатура: от A до Z

Подробное исследование работы клавиатуры ПК и обслуживающих ее прерываний

При работе с любой программой стоит вам только нажать на клавишу, как на экране мгновенно появляется символ. Это позволяет предположить, что между монитором и клавиатурой существует непосредственная связь. В действительности в процесс, сопровождающий нажатие на клавишу, включены мириады составных частей.

Первым шагом на пути к пониманию работы клавиатуры становится знакомство с системой обслуживания прерываний ПК. Под прерыванием понимается временное прекращение центральным процессорным устройством (CPU — ЦПУ) текущей работы для выполнения некоторых посторонних действий, по завершении которых процессор возвращается в прежнее состояние и продолжает прерванную работу. Каждое прерывание имеет свой номер, по которому ЦПУ находит подпрограмму для его обработки. Семейство процессоров 80x86 обеспечивает работу с двумя типами прерываний — программными и аппаратными. Аппаратное прерывание возникает тогда, когда какое-либо устройство нуждается в экстренном обслуживании и, как правило, такое прерывание — большая неожиданность для ЦПУ.

В отличие от аппаратных, программные прерывания инициируются изнутри основной программы при помощи специальных команд процессора. Программное прерывание можно представить так, как будто программа прерывает сама себя для выполнения подпрограммы, не являющейся частью ее кода. Обращение к программному прерыванию похоже на вызов функции, за тем исключением, что нет необходимости написа-

ния самой функции. К одной из основных задач операционной системы (DOS или OS/2) относится обеспечение пользователя набором стандартных подпрограмм обработки прерываний с тем, чтобы отпала необходимость многократного переписывания наиболее общих (и, надо отметить, наиболее неприятных) наборов команд для всех выполняемых прикладных программ.

При нажатии на клавишу электронные схемы клавиатуры генерируют аппаратное прерывание, информирующее ЦПУ о том, что произошло событие, которое необходимо немедленно обработать. Экстренность обработки обусловлена тем, что клавиатура способна запоминать только до 4 нажатий. Если эти нажатия не будут переданы процессору, то они потеряются. Для предотвращения этого ЦПУ запоминает текущее состояние задачи (например, пересчет электронной таблицы) и выполняет специальную программу обработки сигналов клавиатуры. Эта программа называется INT 09H или "обработчик аппаратного прерывания от клавиатуры". Так как обработка данных с клавиатуры относится к наиболее важным функциям системы, программа обработки INT 09H встроена в ПЗУ и является частью BIOS.

Прерывание INT 09H

Основная функция программы BIOS INT 09H — получение информации о нажатии на клавишу через порт ввода/вывода А микросхемы 8255-А, обработка этой информации и сохранение ее в буфере клавиатуры. Буфер клавиатуры — это небольшой участок памяти в зарезервированной области ОЗУ, называемой областью данных BIOS. Единственная информация, которую программа обслуживания прерываний получает от клавиатуры — это число, называемое скан-кодом и представляющее собой физический номер нажатой или отпущенной клавиши. Большинство прикладных программ ожидают поступления от клавиатуры кода символа (ASCII-кода), поэтому каждое нажатие на клавишу приходится специальным образом обрабатывать. Например, при нажатии на клавишу "А" клавиатурой посылается число (скан-код) 1ЕН. Программа INT 09H определяет по таблице, что это клавиша А. Попутно INT 09H проверяет состояние несимвольных клавиш, таких как CapsLock и левый или правый Shift с тем, чтобы определить, прописная это буква или строчная. Затем скан-код и его ASCII-интерпретация запоминаются в буфере клавиатуры и программа INT 09H завершается. ЦПУ — или, вернее, счетчик команд — возвращается на то самое место, в котором его работа была прервана. При этом система восстанавливается таким образом, что прерванная программа и "не подозревает", что была на некоторое время "заморожена".

Если работа INT 09H происходит "совершенно незаметно", то как же осуществляется передача прикладной программе информации о введенных символах? Для этого в операционной системе имеется несколько подпрограмм, объединенных под эгидой обработчика программного прерывания INT 16H.

Прерывание INT 16H

Если единственная обязанность аппаратного прерывания INT 09H — это прием данных с клавиатуры, их интерпретация и занесение в буфер, то программное прерывание INT 16H обладает большей гибкостью, предоставляя целых 3 подфункции (пронумерованные от 0 до 2). Прикладная программа задает необходимую функцию путем передачи обработчику INT 16H номера этой функции в регистре АН перед выполнением прерывания.

Первая функция (с номером 0 или 00H) передает прикладной программе информацию о нажатой клавише (ее скан- и ASCII-коды) из буфера клавиатуры. При использовании этой функции надо учитывать одну ее особенность. Дело в том, что буфер клавиатуры большую часть времени пуст, а прерывание INT 16H 00H (т.е. прерывание INT 16H, функция 00H) обязано вернуть информацию о нажатой клавише. Поэтому функция 00H заставляет ЦПУ выполнять цикл ожидания до тех пор, пока в буфере не появится какой-либо код — т.е. пока не будет нажата какая-нибудь клавиша.

Для того, чтобы понять, каким образом во время пребывания процессора в цикле ожидания в буфер поступает информация, обратимся к прерыванию INT 09H. Это аппаратное прерывание вырабатывается при любом нажатии на клавишу и, прерывая просмотр функцией INT 16H 00H буфера клавиатуры, заносит в него данные. После окончания работы прерывания INT 09H, функция INT 16H 00H внезапно обнаруживает поступившие с клавиатуры данные и передает их прикладной программе.

Функция 01H прерывания INT 16H в некоторых случаях более эффективна. Эта функция позволяет программе определить, есть в буфере данные или нет и затем немедленно возвращает управление прикладной программе. Все, что остается программе — это периодически обращаться к функции INT 16H 01H и проверять заполненность буфера. Если будет получен ответ "да", то программа сможет немедленно прочитать символ с помощью функции INT 16H 00H.

Последняя функция прерывания INT 16H имеет номер 2 (02H) и возвращает текущую информацию о клавишах CapsLock, Insert, NumLock, ScrollLock, Alt, Ctrl, левый и правый Shift. Мы вернемся к этой функции несколько позднее.

Таким образом, главное отличие между INT 09H и INT 16H состоит в том, что INT 09H является аппаратным прерыванием, выполняемым BIOS, и используется исключительно клавиатурой для сохранения данных о нажатых клавишах. INT 16H — это программное прерывание, выполняемое ПЗУ BIOS и используемое прикладными программами по мере необходимости для получения данных о нажатых клавишах.

Подробнее о прерываниях

Так как концепция прерываний является ключевой в понимании функционирования клавиатуры (как впрочем и любых других устройств), рассмотрим работу прерываний поподробнее. Как для программных, так и для аппаратных прерываний семейство процессоров 80x86 выполняет одну и ту же последовательность операций. В обоих случаях управление временно передается программе, называемой обработчиком прерывания. При возникновении аппаратного прерывания переход на его обработчик вызывается воздействием сигнала, поступающего на специальный вывод процессора. Программные же прерывания инициируются самим процессором при выполнении инструкции INT (INTerrupt — прерывание). У команды INT имеется обратная команда — IRET (Interrupt RETurn — возврат из прерывания), используемая обработчиком прерывания для возврата управления прерванной программе. Рассмотрим сначала, что делает ЦПУ при выполнении команды INT.

При возникновении прерывания, независимо от того, инициировано оно извне — аппаратно, или изнутри — программой, ЦПУ прежде всего сохраняет текущее состояние регистра флагов в стеке. После этого очищаются флаг прерываний (IF) и флаг трассировки (TP). Очистка флага IF (Interrupt-enable Flag) делает то же, что и инструкция CLI (Clear Interrupt-enable Flag — очистка IF, запрет прерываний) — запрещает прерывания (все, кроме немаскируемых прерываний, напр., INT 02H), обеспечивая непрерывную работу ЦПУ в течение всего времени выполнения процедурой INT критичного задания. Обработчик прерываний должен разрешить прерывания сразу, как только это станет возможно, используя команду STI (Set Interrupt-enable Flag — установка IF, разрешение прерываний). Тем самым будет разрешено выполнение других критичных по времени прерываний. Примером критичного по времени прерывания может служить ввод символа через порт связи и его буферизация. Однако инструкция STI может быть опущена, если код обработчика очень короткий. Очистка флага TF (Trap Flag) необходима для обеспечения правильной работы программ-трассировщиков. Если флаг трассировки не сбросить, то и сам обработчик будет прерываться после каждой выполненной инструкции, что приведет к банальному зависанию.

Далее ЦПУ сохраняет в стеке текущее значение регистра сегмента кода (CS). Затем в CS заносится значение сегмента адреса обработчика прерывания, взятое из таблицы векторов прерываний. Таблица векторов прерываний находится в нижней части памяти — в самом ее начале (стартовый адрес 0000:0000). Индексация (выборка нужного адреса) в таблице производится путем умножения номера прерывания на 4 и добавления 2 (для сегментного адреса). Двойка добавляется из-за обратного порядка записи адресов, принятого в процессорах Intel — вначале размещается смещение, а затем сегмент. Такое “перевертывание“ является следствием принципа “более значимая информация хранится в более старших адресах“. (На самом деле никакого перевертывания здесь нет. Просто более старшие информационные разряды чисел записывается в более старших байтах памяти. Так адрес FFFC:08A5 реально в байтах памяти запишется, как A5 08 FC FF, аналогично тому, как слово 25B7 записывается в памяти в виде B7 25.)

После этого в стеке сохраняется скорректированное значение указателя инструкций (или счетчика команд — IP — Instruction Pointer), указывающее на следующую за прерванной выполняемую команду. Далее в IP заносится смещение обработчика прерываний из таблицы векторов прерываний. Индексация смещения в таблице производится путем умножения номера прерывания на 4.

В результате этого процесса управление передается процедуре, указываемой новой парой адресов CS:IP, т.е. программе обработки прерывания. (На самом деле производится просто far-вызов программы-обработчика по адресу из таблицы векторов прерываний — CALL FAR (INT*4), где INT — номер прерывания.) Как указывалось, одно из первых действий, которое выполняет обработчик, это разрешение прерываний командой STI. Затем все регистры, которые обработчик может модифицировать, сохраняются в стеке. Далее выполняется непосредственно обработка прерывания. По ее завершении значения регистров восстанавливаются. Если обработчик вызывается аппаратным прерыванием, он информирует Программируемый Контроллер Прерываний (PIC) об окончании обслуживания прерывания посылкой команды (EOI).

И, наконец, обработчик выполняет инструкцию IRET. IRET возвращает управление прерванному процессу, выполняя действия, противоположные тем, которые выполняются инструкцией INT. Первым из стека заполняется регистр IP. Следующее слово выталкивается в регистр сегмента кода — это адрес сегмента, в котором расположена следующая выполняемая инструкция. Последнее слово выталкивается в регистр флагов.

Этот процесс возвращает управление программе точно в то место, где она была прервана. Так как состояние машины (особенно ее регистров и флагов) восстанавливается в исходное, прерываемая программа “не чувствует“ приостановки в работе и продолжает выполнение без каких-либо потерь.

Методика использования таблицы векторов прерываний упрощает внесение изменений в программы прерываний. Все, что нужно сделать — это изменить адрес входной точки прерывания в таблице. Наиболее общие изменения в программах обслуживания прерываний происходят при изменениях в операционной системе. Например, в DOS 3.x вектора прерываний, обслуживающих систему, отличны от DOS 2.x. Аналогично этому, функции обработки прерываний BIOS у IBM PC/AT и клонов будут различаться, но ЦПУ определит правильный адрес в обоих случаях.

Подробно о прерывании INT 09H

Напомним, что все события, связанные с клавиатурой, можно разделить между двумя прерываниями — аппаратным INT 09H, ответственным за буферизацию данных клавиатуры, и программным INT 16H, выбирающим данные из буфера и передающим их прикладной программе. Рассмотрим прерывание INT 09H подробнее (см. рис. 1).

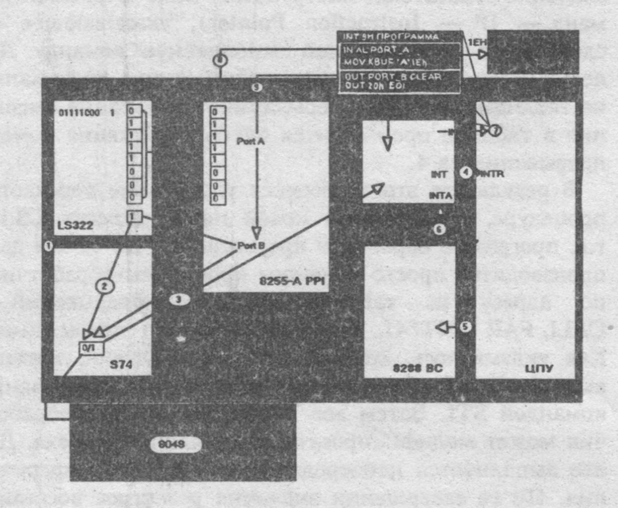


Рис. 1.

Почти во всех клавиатурах содержится микросхема 8048. 8048 является однокристальной микро-ЭВМ, состоящей из 8-разрядного микропроцессора, 64 байт ОЗУ и ПЗУ с содержащимися в нем скан-кодами клавиш и программой. При нажатии или отпускании клавиши 8048 ищет в ПЗУ скан-код, присвоенный этой клавише. Скан-код — это не то же самое, что ASCII-код. Например, для клавиши "А" скан-код равен 1ЕН, а ASCII коды — 41Н для прописной А и 61Н для строчной. За преобразование скан-кода ответственно прерывание INT 09H. Кроме того, 8048 различает этапы нажатия и отпускания клавиш. Отпускание клавиши отмечается установкой в 1 старшего бита скан-кода.

После определения правильного скан-кода 8048 подготавливается к пересылке 1-байтного кода в схему LS322, расположенную на системной плате ПК. (Наименования микросхем в каждой конкретной машине могут быть иными, однако назначение их одинаковое). Скан-код посылается до тех пор, пока процессор не получит "зеленый свет" от схемы S74, расположенной на системной плате, подтверждающий, что последний переданный код был прочитан и расшифрован программой обработки прерывания INT 09H.

Если скан-код не может быть передан непосредственно в данный момент, он заносится в буфер ОЗУ клавиатуры, который я назвал бы буфером скан-кодов. (Не следует путать этот буфер с 16-символьным кольцевым буфером в области данных системы, используемым программой INT 09H для хранения символов.) Если буфер скан-кодов полон, то происходит переполнение и данные теряются. В таком случае 8048 информирует об этом системный блок, посылая специальный скан-код FFH, а прерывание INT 09H указывает пользователю на переполнение буфера клавиатуры звуковым сигналом. Переполнение, однако, происходит довольно редко. Если вам уж так хочется, вы можете искусственно добиться переполнения, нажав несколько клавиш одновременно всей ладонью. В основном сигнал переполнения служит для обнаружения таких стихийных событий, как, например, падение книги на клавиатуру. Однако переполнение буфера клавиатуры может легко произойти в случае, если прикладная программа медленно отрабатывает нажатия на клавиши.

Вернемся к нашему примеру. После того, как клавиатура последовательно передаст скан-код буквы А (1ЕН), он будет принят в системном блоке схемой LS328, являющейся 8-разрядным регистром сдвига (см. шаг 1 и последующие шаги на рис. 1). Скан-код посылается последовательным сигналом. Перед данными следует стартовый бит, т.е. всего посылается 9 бит. Приемный регистр сдвига изначально установлен в 0. (Он был очищен в конце предыдущей процедуры обработки.) При поступлении каждого бита, включая стартовый, освобождается место для приема следующего путем сдвига предыдущих 8 бит регистра на 1 разряд влево. После получения последнего бита данных скан-код вновь преобразуется в исходный 1-байтный вид. В регистр сдвига поступило 9 бит, но он может удерживать только 8, поэтому прием последнего бита данных выталкивает из регистра стартовый бит.

Однако стартовый бит не теряется. Он используется для установки триггера S74 в высокое состояние (2), что переводит линию запроса прерываний (IRQ1) INT 09H в высокое состояние (3). Этот триггер одновременно посылает низкий уровень процессору 8048 на клавиатуре, т.е. дает "красный свет" по той же самой линии данных, по которой был передан скан-код. Это запрещает любую передачу данных клавиатурой и переводит процессор 8048 в режим буферизации, описанный ранее.

Аббревиатура IRQ (дословно — "запрос на прерывание" — Interrupt Request) обозначает сигнал, посылаемый устройством, требующим обслуживания. Для разрешения конфликтов, неизбежно возникающих при одновременных запросах от нескольких устройств, каждому устройству присваивается некоторый уровень приоритета. Считается, что наиболее приоритетное устройство имеет уровень 0 (в IBM PC это системный таймер). Для простоты номер приоритета дописывается к "IRQ". Таким образом IRQ1 означает запрос на прерывание от клавиатуры, имеющей уровень приоритета 1. Для работы с поступающими сигналами IRQ в компьютере есть Программируемый Контроллер Пре-

рываний (PIC) 8259, который и решает на приоритетной основе, какое из заданий обслуживать первым при одновременном поступлении нескольких запросов. PIC имеет восемь линий IRQ. (PIC однако не ограничен восемью запросами. Запросы можно соединять цепью, используя один PIC в качестве ведущего, а остальные в качестве ведомых. В AT и PS/2 используется дополнительный ведомый PIC для подключения дополнительных линий IRQ.) Когда PIC получает запрос, он устанавливает в Регистре Запросов Прерываний (IRR) бит, соответствующий этому запросу. IRR — это 8-разрядный регистр, каждый бит которого используется для отслеживания IRQ, нуждающегося в обслуживании. Арбитр приоритетов PIC использует этот регистр для выбора IRQ с наивысшим приоритетом.

Для примера допустим, что запрос от клавиатуры IRQ1 является единственным, требующим обслуживания. PIC устанавливает высокий уровень на линии INT, передающей запрос на обслуживание клавиатуры путем установки высокого уровня на линии Запроса Прерываний (INTR) ЦПУ. ЦПУ будет поддерживать запрос в том случае, если прерывания были перед этим разрешены инструкцией STI. Обычно прерывания разрешены, и запрещаются инструкцией CLI только тогда, когда программе необходимо выполнить критичное по времени задание, такое как изменение указателя стека (SP) или чтение из порта связи. Само прерывание, как уже говорилось ранее, также запрещает прерывания.

Предположим, что ЦПУ примет прерывание. Тогда оно закончит выполнение текущей инструкции, а затем подтвердит запрос прерывания PIC посылкой 3-битного кода Контроллеру Шины 8288 (BC — Bus Controller), как показано в (*5*). Контроллер Шины обеспечивает проход данных к месту назначения. 3-битный код может управлять 7 типами прерываний BC. Сюда входит и управление сигналом, используемым ЦПУ для приема или записи данных в память или порт. Таким образом происходит разделение между инструкциями IN и OUT, предназначенными для портов, и инструкцией MOV, относящейся к операциям с памятью.

В случае запроса на прерывание 3-битный код, посылаемый ЦПУ, является кодом сигнала подтверждения ("зеленым светом"), посылаемым контроллером шины 8288 назад в PIC 8259A по однобитной линии подтверждения прерываний (INTA) (*6*). Первый сигнал INTA, посылаемый в 8259A, используется для синхронизации, подобно стартовому биту при последовательной передаче. Следующее 3-битное подтверждение от ЦПУ к BC посылает сигнал INTA в 8259 для начала обработки запроса. Далее 8259 устанавливает номер вектора прерываний для ЦПУ (в случае IRQ1 это INT 09H) (*7*). 8259 также устанавливает регистр ISR так, что PIC может отслеживать, какое прерывание в настоящий момент обслуживается.

А как же PIC 8259 узнает, что IRQ1 соответствует номер 9? Все дело в том, что PIC — программируемое устройство, и поэтому начальный код, выполняемый при загрузке компьютера, настраивает его нужным образом. Одним из действий, выполняемых при этом, будет настройка старших *5* бит байта, представляющего номер прерывания, передаваемого в ЦПУ. При формировании номера прерывания младшие 3 бита заполняются номером IRQ. Таким образом можно адресовать 8 векторов от IRQ0 до IRQ7. Единственное неудобство здесь состоит в том, что обработчики аппаратных прерываний, подключенных к одному PIC, должны иметь смежные номера.

В IBM PC старшие *5* бит программируются в 00001B, что при дополнении 3 нулями дает номер обработчика таймерного прерывания (IRQ0). Таким образом, IRQ1, заполняя младшие 3 бита, составляет полный байт равный 00001001B или 09H. Восемь уровней IRQ проецируются на прерывания от INT 08H до INT 0FH.

Очистка Программируемого Контроллера Прерываний обработчиком прерываний

Последнее, что должна выполнять программа обработки прерываний — это выдавать сообщение Программируемому Контроллеру Прерываний (PIC) об окончании прерывания. На ассемблере это выглядит так:

```
CLI              ; Запрет прерываний
MOV   AL,20H     ; Посылка неспецифичного конца
OUT   20H,AL     ; прерывания EOI в порт 20H
.                ; Восстановление всех сохраненных
.                ; регистров
.
IRET             ; Возврат в прерванную программу
```

Подробности приведены далее. Если имеются другие запросы IRQ, ожидающие обработки, то сразу после получения сигнала EOI, PIC посылает ЦПУ запрос на прерывание. Если прерывания разрешены, (любой хороший обработчик прерываний разрешает их, получив управление), то ЦПУ обратится к прерываниям. Это означает, что обработчик прерываний, заканчивающий работу, может быть прерван сразу после инструкции OUT (в нашем примере), до инструкции IRET. Это может показаться безопасным, однако результат будет катастрофическим, если такая ситуация повторится много раз подряд. Каждое дополнительное прерывание увеличивает стек на 6 байт плюс некоторый объем, используемый обработчиком прерываний для сохранения регистров. Многократное вложение прерываний означает неконтролируемый рост стека, что может привести к перезаписи (порче) информации в области данных или кода. В этом случае система будет испорчена.

Что же необходимо сделать для того, чтобы оказалось достаточно времени для выталкивания из стека сохраненных регистров, а также 6 байт (IP, CS, Flags) командой IRET. Для этого надо запрещать прерывания командой CLI непосредственно перед посылкой EOI в PIC. Разрешение прерываний будет выполнено

5 Зак. 2503

командой IRET. Помимо восстановления CS и IP IRET восстанавливает содержимое регистра флагов и делает его таким, каким оно было до прерывания. Так как бит прерываний (бит, переключаемый инструкциями CLI и STI) в регистре флагов перед возникновением прерывания был установлен, то восстановление регистра прерываний разрешает прерывания. Стек будет очищен и далее спокойно смогут выполняться другие прерывания.

Для понимания того, что означает каждый бит байта 20H, посылаемого в порт 20H, необходимо заметить, что схема 8259A имеет много режимов работы, но в ПК она работает в режиме по умолчанию. В этом режиме IRQ выбираются в соответствии с фиксированными приоритетами. Чем меньше номер запроса IRQ, тем выше его приоритет. IRQ0, INT 08H (системный таймер) имеет наивысший приоритет. Запрос от клавиатуры IRQ1, INT 09H имеет второй уровень приоритета. Принтер (IRQ7, INT 0FH) имеет самый низкий приоритет. Прерывание с более низким приоритетом не может произойти во время обработки прерывания с более высоким приоритетом. Регистр обслуживания PIC (ISR) отслеживает, какое из прерываний обслуживается. В соответствии с этой логикой приоритетов PIC считает, что если получен сигнал EOI от обработчика прерываний, то нужно выбрать следующий по приоритету запрос и начать его обслуживать. Здесь заключена некоторая опасность, состоящая в том, что если послать лишний EOI, то он будет отнесен к следующему, еще не обрабатываемому запросу, который будет сброшен.

Младшие 3 бита 8-битного кода EOI, посылаемого в порт 20H, используются PIC для определения уровня приоритета обработчика прерывания, посылающего EOI. Так как PIC автоматически определяет уровень, то этот специфичный тип EOI не нужен и не используется. Поэтому младшие 3 бита устанавливаются в 0. Верхние 3 бита используются для индикации типа EOI; неспецифичный для данного режима тип кодируется как 001B. Средние два бита не используются и всегда устанавливаются в 0. В результате получаем код 00100000B или 20H. Адрес порта 20H — это место подключения схемы 8259A — чисто случайно имеет то же значение, что и посылаемый сигнал EOI.

При вызове прерывания ЦПУ использует номер прерывания как указатель в таблице векторов, содержащей адрес программы прерывания (как это было рассмотрено ранее). Так как таблица векторов расположена внизу памяти, то вектор прерывания INT 00H размещается в четырех байтах, начиная с самого первого байта памяти (0000:0000). Каждый адрес (вектор) занимает 4 байта — 2 для сегмента и 2 для смещения. Для поиска вектора, связанного с прерыванием, ЦПУ умножает номер прерывания на 4 и использует результат как индекс в таблице векторов. Выбранный адрес загружается в пару регистров CS:IP ЦПУ, эффективно передавая управление программе обслуживания прерывания. В случае клавиатурного прерывания — это обработчик INT 09H.

Первое, что выполняет INT 09H — получает скан-код. Вспомним, что скан-код находится в регистре сдвига LS322. INT 09H не имеет прямого доступа к этой схеме и к скан-коду. Но LS322 посылает скан-код по 8 параллельным линиям (а не по последовательной линии) в порт В (адрес 60H) другой микросхемы, называемой Программируемым Параллельным Интерфейсом (PPI) 8255-А.

Помимо обслуживания функций клавиатуры PPI считывает значения переключателей на системной плате компьютера и управляет динамиком. Для прерывания INT 09H PPI служит источником скан-кодов. Программа INT 09H считывает скан-код из порта А, преобразует его в символ ASCII и запоминает скан-код и символ в буфере клавиатуры. Обратите внимание на то, что INT 09H — не единственная программа, способная считывать из порта А. Так как считывание данных из этого порта, в отличие от многих других, не уничтожает их, то любая программа может спокойно считывать скан-код.

После того, как программа INT 09H прочитает скан-код, можно дать "зеленый свет" на посылку нового скан-кода. Это выполняется путем кратковременной установки бита 7 порта В в 1 (см. 9 и рис. 2 — команды на ассемблере). Бит 7 соединен с линией очистки как LS322, так и S74. LS322 обнуляет выходы сдвигающего регистра, подготавливая его к приему следующего скан-кода. S74 сбрасывает линию IRQ1, информируя процессор клавиатуры 8048 о готовности к приему нового скан-кода.

```
IN   AL,20H         ; Ввести текущее состояние порта В
OR   AL,80H         ; Установить старший бит
JMP  $ + 2          ; Переход на задержку
                    ; для быстрых машин
OUT 20H,AL          ; Сброс клавиатуры
AND AL,NOT 80H      ; Сбросить старший бит
JMP  $ + 2          ; Задержка
OUT 20H,AL          ; Восстановить состояние порта В
```

Рис. 2.

Как только скан-код и его ASCII-эквивалент будут сохранены в буфере, а клавиатура получит разрешение работы, программе INT 09H остается только сообщить PIC, что она завершила работу и передает управление прерванной программе (10). Это выполняется посылкой сигнала EOI, закодированного как 20H, в порт PIC 20H. Как отмечалось выше, совпадение номера порта и кода является чисто случайным. Сигнал EOI сбрасывает в регистрах ISR и IRR бит, связанный с IRQ1, освобождая линию запроса. PIC теперь может приступить к обработке аппаратных прерываний с более низким приоритетом, включая и следующий IRQ1.

INT 09H завершает свою работу и передает управление прерванной программе путем выполнения команды IRET. Более подробную информацию по 8259A смотрите в Peripherial Handbook (Intel, 1990).

Скан-коды клавиатуры

Как я уже указывал, INT 09H интерпретирует скан-код, поступающий в порт A PPI 8255A. Фактически программа INT 09H выполняется всякий раз при активизации клавиатуры. Сюда относятся нажатие клавиши, ее отпускание, а также удержание в нажатом состоянии. Клавиатура просто посылает скан-код в ПК программе INT 09H, которая определяет, как его необходимо интерпретировать. Программа INT 09H, расположенная в BIOS, преобразовывает скан-коды, посылаемые клавиатурой, в ASCII-символы, которые могут использовать прикладные программы. Здесь необходимо принять во внимание следующее обстоятель-

Визуальное отображение порта скан-кодов клавиатуры.
Michael J. Mefford

```
        TITLE  PORT-A.ASM
        PAGE   60,132

_TEXT   SEGMENT PUBLIC ´CODE´
        ASSUME     CS:_TEXT
        ASSUME     DS:_TEXT
        ORG     100H
START:  JMP     MAIN

;       DATA AREA

                DB    CR,SPACE,SPACE,SPACE,CR,LF
COPYRIGHT       DB    “PORT-A 1.0 (c) 1990 “
PROGRAMMER      DB    “Michael J. Mefford“,CR,LF,LF,“$“
                DB    CTRL_Z

CR              EQU   13
LF              EQU   10
CTRL_Z          EQU   26
SPACE           EQU   32
BOX             EQU   254
ESC_SCAN        EQU   1
PORT_A          EQU   60H
BIOS_INT_9      DW    ?,?

MENU    LABEL BYTE

DB      “Press and release any key to see “
DB      “make and break scan code“,CR,LF
DB      “Press Esc to Exit“,CR,LF,LF,“$“

;       CODE AREA

MAIN    PROC  NEAR
 CALL   CLS                      ; Очистка экрана
 MOV    DX,OFFSET COPYRIGHT      ; Вывод Copyright и меню
 MOV    AH,9
 INT    21H
 MOV    DX,OFFSET MENU
 INT    21H
 MOV    AX,3509H                 ; Получить прерывание
                                 ; клавиатуры

 INT    21H
 MOV    BIOS_INT_9[0],BX         ; Сохранить старое
 MOV    BIOS_INT_9[2],ES         ; прерывание
 MOV    DX,OFFSET PORT_A_INT_9   ; Установить новый
 MOV    AX,2509H                 ; вектор
 INT    21H

; Находиться в цикле до появления Esc
GET_KEY:
 XOR    AH,AH                    ; Ожидать нажатия на клавишу
 INT    16H
 CMP    AH,ESC_SCAN              ; Если Esc, выход
 JNZ    GET_KEY                  ; Иначе — продолжить
EXIT:
 MOV    DX,BIOS_INT_9[0]         ; Восстановить старый
 MOV    DS,BIOS_INT_9[2]         ; вектор INT 9
 MOV    AX,2509H
 INT    21H
 CALL   CLS                      ; Очистить экран.
 MOV    AX,4C00H                 ; Выход с error level = 0
 INT    21H

MAIN    ENDP

;       ПОДПРОГРАММЫ

; Данная процедура вызывается каждый раз при нажатии
; на клавишу и ей передается значение, что позволяет
; взглянуть на скан-код и отобразить его

PORT_A_INT_9 PROC  NEAR
 ASSUME  DS:NOTHING
 PUSH AX                         ; Сохранить регистры
 PUSH BX
 PUSH CX
 IN     AL,PORT_A                ; Получить скан-код
 CALL   HEX_OUTPUT               ; и отобразить его
OLD_INT_9:
 POP    CX                       ; Восстановить регистры
 POP    BX
 POP    AX
 JMP    DWORD PTR BIOS_INT_9  ;переход на BIOS INT 9
PORT_A_INT_9  ENDP

;----------------------------;
HEX_OUTPUT  PROC  NEAR
 MOV  BX,AX                      ; Сохранить число в BX
 MOV  CX,204H                    ; 4 позиции/слово; 4 бита/символ
ROTATE_HEX:
 ROL   BL,CL                     ; Старшие биты в младшие
 MOV   AL,BL                     ; Сохранить число в AL
 AND   AL,1111B                  ; Маскировать все, кроме четырех
 ADD   AL,“0“                    ; Преобразовать в ASCII
 CMP   AL,“9“                    ; Буква?
 JLE   PRINT_HEX                 ; Если нет — печать
 ADD   AL,7                      ; Иначе — подстроить
PRINT_HEX:
 MOV   AH,0EH                    ; Печать через BIOS
 INT   10H
 DEC   CH                        ; Все сделано для этих позиций?
 JNZ   ROTATE_HEX                ; Если нет, взять следующую
 MOV   CX,2                      ; Вывести два пробела
DELIMIT:                         ; между скан-кодами
 MOV   AL,SPACE                  ; в качестве разделителей
 MOV   AH,0EH
 INT   10H
 LOOP  DELIMIT
 RET
HEX_OUTPUT  ENDP

;----------------------------;
CLS        PROC  NEAR
 MOV    AH,0FH                   ; Текущий видео режим
 INT    10H
 CMP    AL,7                     ; Моно?
 JZ     CLEAR_SCREEN             ; Если да — очистить экран
 MOV    AL,3                     ; Иначе проверить,
                                 ; что текстовый
CLEAR_SCREEN:
 XOR    AH,AH                    ; Очистить экран
 INT    10H                      ; установкой режима
 MOV    AX,500H                  ; Убедиться, что страница
 INT    10H                      ; нулевая
 RET

CLS        ENDP
_TEXT   ENDS
        END             START
```

Рис. 3.

ство. INT 09H является аппаратно-зависимой функцией, поэтому программа для 83-клавишной клавиатуры отличается от версии программы для расширенной 101-клавишной клавиатуры. Очевидно, что для связи букв с дополнительными клавишами требуются дополнительные коды.

Для лучшего понимания обратимся к программе PORT-A.ASM (рис. 3), которая в реальном времени отображает скан-коды, посылаемые при нажатии и отпускании клавиш (клавиатура посылает скан-коды в порт А PPI, адрес 60H.) Скан-коды считываются программой PORT-A до того, как программа INT 09H доберется до них, путем перехвата прерывания 09H. Перехват прерывания означает замену адреса прерывания в таблице векторов прерываний на адрес своей программы обработки. Исходный адрес при этом запоминается и, как только программа PORT-A или любая другая перехватившая программа завершится, управление может быть передано первоначальному обработчику прерываний. (Хотя перехват прерываний похож на передачу управления TSR-программе при нажатии "горячей клавиши", PORT-A.ASM не является TSR-программой.) При нажатии клавиши Esc программа PORT-A восстанавливает прежнее значение в таблице векторов прерываний и возвращается к приглашению DOS. Остальные подробности программы описаны в комментариях в листинге программы. Вы можете скомпилировать представленный код.

После запуска программы PORT-A при любом нажатии на клавишу на экране появляется скан-код этой клавиши. При отпускании этой клавиши на экране появляется другой скан-код. Для 83-клавишной клавиатуры код отпускания равен коду нажатия плюс 80H (10000000B), что устанавливает старший бит скан-кода. Это также верно и для большинства клавиш расширенной клавиатуры, однако для новых (дополнительных) клавиш — специализированных клавиш курсора и некоторых комбинаций клавиш — посылается серия скан-кодов, некоторые из которых имеют старший бит, установленный в 1 даже при нажатии на клавишу. До появления расширенной клавиатуры пользователи могли легко отличить нажатие клавиши от отпускания проверкой старшего бита скан-кода. Этот способ не работает с расширенной клавиатурой. Программа PORT-A может стать удобным инструментом для определения скан-кодов, которые должна обрабатывать программа INT 09H при нажатии на ту или иную клавишу. Работа программы PORT-A вполне понятна, однако значения выводимых скан-кодов поначалу кажутся весьма странными. Все становится ясно после понимания работы INT 09H. Для этого рассмотрим логику декодирования программы INT 09H шаг за шагом.

Скан-код, считываемый программой INT 09H из порта А, в большинстве случаев является числом, представляющим собой относительное положение клавиши на клавиатуре. Так, скан-код 01H относится к клавише Esc, 02H — к клавише !/1, 03H — @/2 и т.д. слева направо по верхнему ряду клавиатуры. В отличие от большинства компьютерных схем нумерации, отсутствует клавиша, связанная с кодом 0. INT 09H использует 0 как псевдо-скан-код для отображения комбинации Ctrl-Break. После клавиши BackSpace последовательная нумерация возобновляется с клавиши Tab в ряду QWERTY. После перечисления клавиш пишущей машинки нумерация продолжается на функциональных клавишах, а далее — на дополнительных. Некоторые клавиши, такие как Esc и функциональные, на разных клавиатурах расположены в разных местах, поэтому для них система нумерации не имеет смысла. Для сохранения совместимости скан-коды остаются теми же. Помните, что скан-код не имеет ничего общего с ASCII-эквивалентом клавиши.

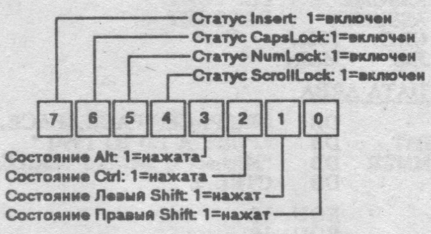


Рис. 4,

Для дешифровки скан-кодов INT 09H содержит специальную таблицу поиска. Для поиска соответствия в таблице используется ряд алгоритмов. Как только скан-код расшифрован, INT 09H сохраняет скан-код и ASCII-символ в буфере клавиатуры и возвращает управление прерванной программе. Однако до передачи управления INT 09H проверяет, не является ли скан-код кодом FFH. Как упоминалось выше, код FFH является кодом клавиатуры, указывающим на переполнение. При обнаружении такого кода INT 09H сообщает об этом сигналом динамика. В противоположном случае, полагая, что переполнение отсутствует, INT 09H просматривает таблицу перевода в логическом порядке. Сначала проверяются клавиши смены функций: клавиши сдвига — левый и правый Shift, Ctrl, Alt и четыре фиксируемые клавиши — Insert, CapsLock, NumLock, ScrollLock. Если скан-код принадлежит одной из четырех клавиш сдвига, это отображается установкой соответствующего бита в информационном байте KD_FLAG, расположенном по адресу 40:17H в области данных BIOS (рис. 4). При обнаружении скан-кода отпускания клавиши сдвига соответствующий бит сбрасывается в 0. Такой метод позволяет программе INT 09H всегда знать, какая из этих клавиш нажата.

Клавиши сдвига

Лучший способ проиллюстрировать логику работы клавиш сдвига — это посмотреть, что происходит до и после расшифровки скан-кода программой INT 09H. Сначала рассмотрим, как INT 09H отработает нажатие нескольких клавиш в командной строке DOS.

Нажмем и будем удерживать левую клавишу Shift. Внешне ничего не происходит. Продолжая удерживать левый Shift, нажмем клавишу "A". Как и ожидалось, появится прописная буква A (полагаем, что клавиша CapsLock не включена), а вслед за ней — целый ряд прописных A, возникающих вследствие работы функции автоповтора клавиатуры. Отпустим левый Shift, продолжая удерживать клавишу "A" нажатой. Прописная A заменится на строчную а, а функция автоповтора продолжит свою работу.

При помощи программы Port-A можно увидеть, что в действительности происходит со скан-кодами. Нажмите Esc для отмены ничего не значащей команды AAAAAAaaaa и запустите программу PORT-A. После этого повторите те действия, которые вы выполняли в командной строке DOS — начните с нажатия и удержания левой клавиши Shift. Появится и будет повторяться скан-код 2AH. (Клавиатура начинает автоповтор примерно через 0,5 секунды, если не изменена задержка. Это справедливо для всех клавиш, включая и клавишу Shift.) Продолжая удерживать клавишу Shift нажатой, нажмите клавишу "A". Поток кодов 2AH заменится на 1EH — скан-код клавиши "A". (Автоповтор прекратится, как только клавиатура обнаружит, что нажата другая клавиша. Если удерживать и вторую клавишу, то начнется автоповтор этой клавиши.)

Далее отпустите клавишу Shift, но не клавишу "A". В потоке скан-кодов 1EH, принадлежащих клавише "A", появится один код AAH, а затем возобновится поток кодов 1EH. Обратите внимание на то, что в отличие от работы в DOS, где поток прописных A заменяется на поток строчных а, в данном случае отпускание клавиши Shift не изменяет скан-кода. Отпустите клавишу "A"; вывод закончится кодом 9EH — отпускание клавиши (1EH+80H = 9EH).

Проверка байтов Статуса Сдвига и Состояния Сдвига

(Перед выполнением этих упражнений удалите из файла AUTOEXEC.BAT все резидентные программы обслуживания клавиатуры, например, DOSEDIT.)

Для того, чтобы увидеть изменение KD_FLAG, запустите отладчик DEBUG и по подсказке в виде знака минус введите:

D 40:17 L1 (дамп сегмента 40H, смещение 17H, длина 1 байт)

На экране появится состояние байта KD_FLAG (см. рис. 4) Описанные ниже действия будет проще выполнить, если его значение равно 0. Если оно не равно 0, то это означает, что нажата одна из клавиш сдвига. При необходимости переключите клавиши сдвига (Insert, CapsLock, NumLock и/или ScrollLock) и повторите приведенную выше команду. В конце концов вы увидите 0. Во время работы с DEBUG можно использовать клавишу F3 для повтора последней команды.

Нажмите F3 для отображения команды D 40:17 L1, но пока не нажимайте Enter. Сначала нажмите Insert для включения режима вставки и после этого нажмите Enter. Вы увидите значение 80H, указывающее на то, что режим вставки активен. Снова нажмите Insert для его деактивации, затем по одной нажимайте другие фиксируемые клавиши сдвига (CapsLock, NumLock, ScrollLock). Нажимая F3 и Enter после каждого из переключений, наблюдайте производимый эффект. Проверьте, соответствуют ли полученные значения приведенным на рис. 4.

Далее нажмите CapsLock и NumLock. Нажмите F3 и Enter. На этот раз вы увидите значение, являющееся комбинацией отдельных битов, а именно, 60H (40H+20H). Заметьте, что указанным способом нельзя наблюдать состояние клавиш Ctrl и Alt в байте KB_FLAG, так как комбинации Ctrl-Enter и Alt-Enter не генерируют возврата каретки.

Теперь проделайте подобные операции с байтом состояния клавиш KD_FLAG1, расположенным по адресу 40:18 в области данных BIOS. Введите:

D 40:18 L1

На этот раз вы увидите 0, независимо от состояния фиксируемых клавиш. KD_FLAG1 используется BIOS для отслеживания нажатия нескольких клавиш одновременно. Для того, чтобы это увидеть, повторите предыдущую команду, не нажимая клавиши Enter, затем нажмите и удерживайте клавишу Insert и нажмите Enter. Вы увидите значение 80H, указывающее на то, что клавиша Insert находится в нажатом состоянии. Проэкспериментируйте с другими клавишами сдвига, проделывая эту же процедуру. Для выхода из отладчика введите Q.

Как можно заметить, программа INT 09H интерпретирует скан-коды в виде строчных и прописных букв. В действительности, при нажатии на левый Shift программа INT 09H получает код 2AH и устанавливает (оператором OR) бит левой клавиши Shift (бит 1) в байте KD_FLAG. Та же логика используется и при автоповторе клавиши Shift. Последовательная установка одного и того же бита не приводит к каким-либо изменениям. Если нажимается клавиша "A", то INT 09H обнаруживает скан-код 1EH, соответствующий строчной букве а. В то же самое время INT 09H проверяет состояние двух клавиш Shift, просматривая содержимое байта KD_FLAG. Программа обнаруживает, что левый Shift еще нажат, поэтому строчная а становится прописной A.

Прежде чем запомнить A в буфере клавиатуры, INT 09H проверяет бит, соответствующий состоянию клавиши CapsLock (7) в байте KD_FLAG. Если CapsLock выключен, то INT 09H заносит в буфер прописную A. Если же CapsLock включен, то A снова становится строчной а, тем самым отменяя CapsLock.

Далее, при отпускании левой клавиши Shift INT 09H обнаруживает скан-код AAH (AAH = 2AH+80H) и сбрасывает бит левой клавиши

Shift в байте KD_FLAG. При поступлении следующего кода 1EH INT 09H обнаруживает, что левый Shift больше не нажат (а также не нажаты ни правый Shift, ни CapsLock) и записывает в буфер строчную а.

Та же логика смены функций используется и при обработке нажатий клавиш цифровой клавиатуры. Если включен NumLock, то нажатие левой или правой клавиши Shift перед нажатием клавиши на цифровой клавиатуре отменяет установленный статус, тем самым превращая клавиши цифр в клавиши управления курсором. Такая гибкость особенно полезна для пользователей 83-клавишной клавиатуры, которые обычно оставляют NumLock включенным. Когда необходимо использовать цифровую клавиатуру для управления курсором, то вместо переключения NumLock достаточно нажать левый Shift и удерживать его.

У 101-клавишной клавиатуры наряду с блоком цифровой клавиатуры имеются специализированные клавиши управления курсором. Машины с такими клавиатурами обычно при загрузке включают NumLock, поэтому клавиатура готова для ввода чисел. Однако некоторым пользователям трудно побороть привычку и они продолжают использовать цифровую клавиатуру для управления курсором. Так как при загрузке устанавливается NumLock, то при первой попытке управления курсором на экране появляется строка цифр. Для 83-клавишной клавиатуры наблюдается обратная ошибка. Однако можно включать эти клавиши из batch-файла, например из AUTOEXEC.BAT (см. рис. 5).

```
RCX
C
A
MOV   AX,0040
MOV   DS,AX
XOR   Byte Ptr [0017],80
INT20

N INSERT.COM
W

E 109 40
N CAPSLOCK.COM
W

E 109 20
N NUMLOCK.COM
W

E 109 10
N SCROLOCK.COM
W

Q
```

Рис.5.

Вы можете создать файл, текст которого приведен ниже, и направить его в отладчик командой

Debug < TOGGLE.SCR

либо ввести инструкции непосредственно, находясь в отладчике. Результирующие файлы INSERT.COM, CAPSLOCK.COM, NUMLOCK.COM и SCROLOCK.COM можно использовать в пакетных файлах для включения того или иного состояния. Для некоторых расширенных клавиатур данные программы не будут включать соответствующие светодиодные индикаторы, однако результирующее состояние будет верным.

Клавиши-переключатели

Логика работы клавиш-переключателей (Insert, CapsLock, NumLock и ScrollLock) отличается от логики работы клавиш сдвига. Когда программа INT 09H встречает скан-код клавиши-переключателя, она подготавливается к установке (с использованием оператора OR) соответствующего бита в байте KD_FLAG1. Так как все биты в KD_FLAG уже заняты, то для хранения состояния этих клавиш используется второй байт (см. рис. 6). Перед установкой этого бита INT 09H проверяет, не был ли он установлен ранее.

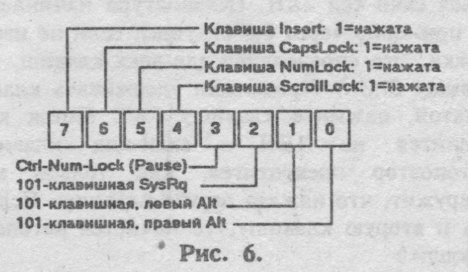


Рис. 6.

Если был, то INT 09H считает, что произошел автоповтор, и не предпринимает никаких действий. С помощью программы PORT-A можно увидеть, что нажатие или отпускание CapsLock действует точно так же, как нажатие и отпускание любой другой клавиши, при этом так же посылается набор скан-кодов в случае автоповтора. Если не использовать дополнительную логику переключения, то автоповтор приведет к непредсказуемым и нежелательным переключениям. Если бит еще не установлен, то он устанавливается, указывая на то, что клавиша нажата. Также переключается бит в KD_FLAG для индикации смены состояния (это выполняется операцией XOR). Для просмотра содержимого KD_FLAG в вашем компьютере воспользуйтесь рекомендациями, приведенными выше.

Набор символов через Alt

Программа INT 09H специальным образом обрабатывает комбинации клавиши Alt с клавишами цифровой клавиатуры. Если клавиша Alt нажата (INT 09H определяет это по биту 3 в KD_FLAG), и если обнаруживается скан-код цифровой клавиши, то INT 09H добавляет число, нарисованное на клавише, к байту по адресу 40:19 в области данных BIOS — ALT_INPUT. Предыдущее значение ALT_INPUT умножается на 10 (для сдвига десятичной точки вправо на одну позицию) прежде, чем будет добавлено новое число. При отпускании клавиши Alt полученное таким образом число интерпретируется как десятичный эквивалент ASCII-кода и помещается в буфер клавиатуры. Для примера, в командной строке DOS нажмите клавишу

Alt и, удерживая ее, нажмите и отпустите клавишу 6, а затем 5 на цифровой клавиатуре. Отпустите клавишу Alt — на экране появится буква A, ASCII-код которой равен 65. При отпускании Alt INT 09H сбрасывает ALT_INPUT в 0. Если при отпускании Alt значение ALT_INPUT равно 0, то ввод не производится, что объясняет невозможность ввода нуля (ASCII — ноль) с помощью этого метода.

Этот альтернативный метод ввода работает только с цифровой клавиатурой, но не с числовыми клавишами в верхнем ряду. Состояние NumLock не имеет значения. Метод набора через Alt используется для ввода символов управления и псевдографики.

Состояние удержания

Если не нажаты клавиши сдвига или клавиши-переключатели, INT 09H проверяет бит 3 в байте KD_FLAG1 для определения активности состояния паузы. Если этот бит установлен, что подразумевает пребывание машины в "замороженном" состоянии, то он сбрасывается и INT 09H прекращает работу. Бит 3 устанавливается нажатием комбинации Ctrl-NumLock на 83-клавишной клавиатуре, или специализированной клавиши Pause на 101-клавишной клавиатуре. Для "замораживания" машины INT 09H зацикливается сразу после установки бита паузы, ожидая его сброса. Сброс осуществляется при нажатии на любую клавишу, за исключением клавиш сдвига. Это происходит потому, что проверка состояния клавиш сдвига выполняется раньше, чем проверка состояния бита паузы.

Для примера, дайте команду DOS DIR, а затем, во время вывода каталога на экран, нажмите Ctrl-NumLock на 83-клавишной клавиатуре или Pause на 101-клавишной для остановки вывода (желательно, чтобы каталог был большим). Теперь нажмите любую клавишу сдвига. Состояние изменится, однако DOS не возобновит вывод списка каталога.

Комбинация клавиш Ctrl-S тоже останавливает вывод каталога, однако это реализовано не программой INT 09H. Ctrl-S обрабатывается и вводится в буфер клавиатуры точно так же, как и любой другой символ, а DOS уже интерпретирует ее как команду паузы.

Ctrl-Alt-Del

Комбинация Ctrl-Alt-Del имеет специальное назначение. При ее обнаружении в ячейку RESET_FLAG, находящуюся в области данных BIOS, заносится значение 1234H и выполняется переход на процедуру инициализации, которая выполняется обычно при включении компьютера. Программа инициализации проверяет RESET_FLAG и пропускает самопроверку в том случае, если его значение равно 1234H. Таким образом осуществляется "теплая" перезагрузка.

Если вы обратитесь к байту KD_FLAG, то заметите, что в нем зарезервированы биты для индикации состояния клавиш Ctrl и Alt, но не Del. Бита состояния клавиши Del нет также и в KD_FLAG1. Это означает, что INT 09H не имеет возможности хранения состояния клавиши Del. Поэтому для перезагрузки необходимо сначала нажать клавиши Ctrl и Alt а затем Del. Порядок нажатия клавиш Ctrl и Alt не имеет значения, важно только нажать Del в последнюю очередь.

Ctrl-Break

При обнаружении этой комбинации INT 09H выполняет несколько действий. Сначала очищается буфер клавиатуры путем установки указателя начала буфера равным указателю конца. Затем INT 09H устанавливает в 1 старший бит байта BIOS_BREAK с адресом 40:71H в области данных BIOS. Любая программа может проверить значение этого бита для определения состояния Ctrl-Break. Затем INT 09H вызывает прерывание INT 1BH. Обычно INT 1BH указывает на обработчик DOS, но любая программа может перехватить этот вектор и тем самым самостоятельно обрабатывать Ctrl-Break. Если прерывание INT 1BH вызывает обработчик DOS, то последний устанавливает внутренний флаг Ctrl-Break DOS. При этом любая программа, использующая функции ввода/вывода DOS при взведенном флаге Ctrl-Break (Ctrl-Break DOS а не INT 09H), будет прервана. DOS отобразит на экране значок "^C" и выведет приглашение командной строки. Последнее, что делает INT 09H с Ctrl-Break — это помещает нулевой символ и скан-код в буфер клавиатуры.

Ввод Ctrl-C тоже интерпретируется как Ctrl-Break. Выполнение этой команды, так же как и выполнение Ctrl-S, не является функцией INT 09H. Ctrl-C обрабатывается INT 09H в той же последовательности, как и любой другой символ, а уже DOS сама интерпретирует ее как команду прерывания.

Print Screen

При нажатии комбинации Shift-PrtScrn на 83-клавишной клавиатуре или специальной клавиши PrintScreen на 101-клавишной, INT 09H просто вызывает INT 05H для посылки дампа экрана на принтер. INT 09H непосредственно печатью не занимается.

Алфавитно-цифровые клавиши

Если INT 09H не обнаруживает специальных скан-кодов, то она полагает, что принятый скан-код принадлежит алфавитно-цифровому символу ASCII. Для алфавитно-цифровых символов INT 09H проверяет состояние CapsLock. Затем INT 09H заносит в буфер как сам ASCII-символ, так и скан-код нажатой клавиши. Это дает возможность прикладной программе либо прочитать ASCII-символ, либо определить нажатую клавишу. Однако, прежде чем что-либо занести в буфер программа INT 09H должна проверить, есть ли в нём место. Если места недостаточно, то INT 09H сообщает об этом сигналом динамика и завершает свое выполнение без занесения символа в буфер.

М. Меффорд

(Окончание следует)

Страна по имени BORLAND

Сегодня специальный выпуск "КомпьютерПресс" впервые посвящен продуктам одной-единственной фирмы. Но диапазон ее деятельности настолько широк, что мы считаем себя просто не вправе не уделить должного места ей в своем журнале, и имя этой фирмы — Borland.

И как только ухитряется Филипп Кан со своей командой настолько диверсифицировать свою деятельность, чтобы производить и превосходные оболочки, и электронные таблицы, и базы данных, и компиляторы — практически все, что только нужно пользователю персонального компьютера. А чего стоит последняя покупка Borland, ни мало, ни много — сама Ashton-Tate, в устойчивости которой, несмотря на известную неудачу с dBASE IV, не сомневался практически никто.

Пять статей, включенных в этот выпуск, посвящены соответственно истории и современной стратегии фирмы, объектно-ориентированному программированию на Turbo Pascal, версии Turbo Pascal, работающей в среде Windows, системе управления базой данных Paradox и генератору отчетов ObjectVision.

Что еще хотелось бы сказать по поводу этого спецвыпуска. Все описываемые здесь продукты давно и весьма успешно продаются рядом совместных предприятий в СССР и, что примечательно, за советские рубли по коэффициенту, который значительно ниже рыночного. А сейчас по вполне достоверной информации Borland, наконец-то, открыла собственное представительство в Москве, и в ближайшем будущем выйдет на советский рынок без посредников.

А теперь, знакомьтесь — Страна по имени Borland.



Корпорация Borland International (США), один из крупнейших производителей программных средств для персональных компьютеров в мире, активно работает на советском рынке с 1990 года. Количество официально закупленных в СССР программ Borland измеряется тысячами, а число нелегально используемых копий — миллионами. Среди партнеров Borland в СССР крупнейшие совместные предприятия: "Интерквадро", концерн "Новинтех", А/О "Диалог-МИФИ", объединение КАТ и некоторые другие фирмы.

Рынок программных средств в СССР:

мирное наступление Borland

Программисты и пользователи часто относятся к программам, как к живым существам: наделяют их "характером", "привычками", серьезно оценивают внешний вид и "поведение". Особенно часто характеристики из этого ассоциативного ряда применяются к наиболее популярным, "нравящимся" программам. На протяжении последних лет многократно приходилось слышать антропоморфные эпитеты в адрес программного обеспечения, поставляемого американской корпорацией Borland (Borland International, Inc.). В чем причины такого отношения? Почему практически на каждом персональном компьютере в СССР можно обнаружить тот или иной продукт Borland? Как, не впадая в апологетику Borland, объяснить тот факт, что опросы, проводимые в СССР, постоянно показывают наивысший интерес советских программистов именно к изделиям фирмы Borland?

В предлагаемом Вашему вниманию спецвыпуске делается попытка ответить на эти вопросы, объяснить и проиллюстрировать отношение потребителей к продукции фирмы. В данной статье анализируются некоторые причины технологического и коммерческого успеха корпорации Borland, дается ретроспектива деятельности Borland в СССР.

Borland: воплощение "американской мечты"

Чтобы понять феномен Borland на советском рынке, следует вернуться в 1983 год — в этом году молодой и предприимчивый математик, Филипп Кан (Philippe Kahn), предложил на американский рынок программных средств язык программирования Pascal по цене $49.95 — на порядок меньше, чем у известных фирм-конкурентов. В этом продукте был реализован целый ряд оригинальных идей, облегчавших труд программиста. Спрос превысил все ожидания! Программный продукт, получивший название Turbo Pascal, стал первым громким коммерческим успехом созданной

Каном корпорации, которая получила название Borland International (в дальнейшем — Borland).

К середине 80-х определились три стратегических направления деятельности Borland: языки программирования, системы управления базами данных, электронные таблицы.

В области языков программирования Borland штурмом взял рынок программных средств, выпуская по неслыханно низким ценам все более и более совершенные версии Turbo Pascal. Для большинства потребителей оказалось дешевле закупить легальную копию продукта, чем раздобыть похищенные копии программного обеспечения, а потом ломать себе голову над тем, где бы скопировать документацию.

Такая коммерческая тактика Borland сохранилась и для других разработок, с той лишь только разницей, что в каждое новое изделие вкладывалось беспрецедентное количество технологических новаций. Количество проданных во всем мире копий языков программирования семейства Turbo (Turbo Pascal, Turbo C, Turbo Assembler) измеряется в настоящее время миллионами!

К 1990 году единственным серьезным конкурентом Borland на рынке языков программирования можно было считать, пожалуй, корпорацию Microsoft. Но в лабораториях Borland уже было разработано секретное оружие: набор программных средств, основанный на объектно-ориентированном программировании (ООП). Этот подход к созданию программных средств эволюционировал, начиная с 60-х годов (SmallTalk, Simula). Его основные преимущества сводятся к многократному повышению производительности труда программистов, особенно, — ведущих коллективные разработки. Исходные тексты программ, разработанных на основе ООП, могут легко обобществляться разработчиками. Упрощается и процесс развития и дополнения действующих программ новыми возможностями. По сути дела, исходный текст программы приобретает черты по-настоящему отчуждаемого продукта, т.е. такого, который может быть легко передан коллеге, партнеру, подчиненному для использования и развития. Сам подход ООП известен достаточно давно, но "трудолюбивыми японцами", первыми освоившими эту прогрессивную технологию, оказались специалисты Borland. В области объектно-ориентированных средств программирования у Borland серьезных конкурентов практически не оказалось.

Для решения задач по обработке электронных таблиц корпорация Borland выпустила продукт Quattro Pro, который по своим функциональным характеристикам превосходит самые свежие версии основных программ-конкурентов. Кроме того, Quattro эффективно работает и на морально устаревших компьютерах невысокой производительности, даже совместимых с IBM PC/XT (внимание, советские руководители!). По числу установленных копий Quattro Pro конкурирует только с продуктами давно захватившего лидерство на этом участке рынка семейства Lotus 1-2-3 фирмы Lotus Development.

Во второй половине 80-х годов начало создаваться впечатление, что Borland обладает даром, сравнимым с прикосновением Мидаса: практически каждый программный продукт, поставляемый фирмой, превращается в золото — завоевывает ведущее место среди лучших изделий для своей категории. Так например, система управления базами данных Paradox фирмы Borland в 1990 году опередила по количеству продаж в США сверхпопулярный в прошлом продукт dBASE корпорации Ashton-Tate. Технологическая победа в этом случае была увенчана и победой стратегической: в июле 1991 года было объявлено о слиянии Borland и Ashton-Tate. Корпорация Ashton-Tate перестает существовать, а все права на ее продукты и разработки передаются Borland. И это при том, что годовой оборот у Borland в 1990 финансовом году был меньше, чем у Ashton-Tate! Коммерческая смелость и технологическое новаторство перевесили арифметическое превосходство капитала.

Таким образом, по всем стратегическим направлениям своего продвижения Borland сумел добиться либо подавляющего, либо заметного преимущества по отношению к продукции традиционных лидеров индустрии программного обеспечения. Количественное выражение этого: теперь, в 1991 году, по суммарному обороту среди фирм — производителей программного обеспечения, ориентированного на IBM-совместимые персональные компьютеры, Borland (440 млн. долл.) уступает только Microsoft (1.5 млрд. долл.) и Lotus (около 600 млн. долл.). За последние два года отмечается двукратный рост прибыли Borland! Это успех, который может привести и к изменению порядка в "большой тройке". Отметим, что корпорации Autodesk и WordPerfect, имеющие обороты свыше 500 млн. долл., здесь нами не рассматриваются, ввиду специфичности предметной области, в которой они ведут разработки.

Если же учесть присущие изделиям Borland исключительно высокие качество, надежность и эстетичность, то сложившуюся ситуацию остается охарактеризовать словами: "Borland идет!".

Borland в СССР: воплотятся ли наши мечты?

С первых же шагов на советском рынке корпорация Borland повела себя в присущем себе стиле: помимо высокого авторитета программных средств (Turbo Pascal, Turbo C++, Paradox, Quattro Pro, ObjectVision) появились весомые дополнительные доводы для привлечения покупателей к продукции фирмы. Эти доводы можно свести к тому, что можно назвать основными принципами маркетинговой политики Borland на советском рынке.

Принцип первый: торговый паритет. Продажа программных средств ведется без ограничений по номенклатуре — самые свежие версии, обладающие высшей технологической новизной даже на рынке США, предлагаются одновременно для всех потребителей в СССР. Залежалого товара у Borland не бывает!

Принцип второй: учет покупательной способности. Приобрести все программы Borland можно и за рубли, и за доллары, без "аннексий и контрибуций" в пользу долларовой номенклатуры. Рублевая цена оп-ре-де-ля-ет-ся в соответствии с покупательной спо-соб-но-с-тью молодого советского рынка, с учетом возможностей малых предприятий, частных фирм и индивидуалов. При этом, средний переводной коэффициент "по курсу Borland" близок к 10 рублям за один доллар цены продукта в США!

Принцип третий: инвестиции в будущее. Заработанные рубли расходуются на территории СССР, в первую очередь, на локализацию программных средств, проведение научно-технических мероприятий, рекламу, обучение и подготовку кадров. Так например, уже в 1991 году начались поставки наиболее популярной СУБД Paradox в "русифицированном" варианте. Разработка, предпринятая известным поставщиком адаптированных версий программного обеспечения зарубежных фирм, СП "ПараГраф" (г. Москва), позволяет отечественным пользователям работать с Paradox на русском языке уже сегодня! Аналогичные работы завершаются для электронных таблиц Quattro Pro.

Принцип четвертый: льготы пользователям конкурентных продуктов (competitive upgrade). Для пользователей программ-конкурентов объявлена "амнистия": при переходе на соответствующий продукт Borland они автоматически получают скидку до 60%, то есть, обладают теми же правами, что и зарегистрированные владельцы предыдущей версии программного средства Borland.

Принцип пятый: союз с потребителем. Особое внимание в СССР корпорация Borland уделяет формированию и поддержке движения пользователей, созданию структур, поддерживающих и развивающих связи фирмы-производителя и потребителей программной продукции.

Действительно, любой потребитель программной продукции (пользователь) склонен персонифицировать применяемые им средства. Во многом это связано с интеллектуализацией процессов обработки информации на компьютере, что неотвратимо закладывает во взаимодействие "человек-программа" глубокую эмоциональную основу. По-видимому, это обстоятельство учитывается не только при проектировании диалоговых средств Borland, которые славятся своей интуитивностью и наглядностью, но и в определении тактики фирмы в отношении связей с общественностью (Public relations). Так например, Borland проводит ежегодный международный семинар пользователей, на котором ведущие специалисты фирмы, начиная с самого Филиппа Кана, рассказывают о своих разработках, выясняют тонкости восприятия их продукции со стороны пользователей, попросту — советуются с ними о перспективах развития отдельных пакетов программ.

Пользователи Borland в СССР, объединяйтесь!

В 1991 году одним из наиболее очевидных успехов Borland в СССР из области связей с общественностью стало создание корпорацией совместно с инициативной группой из числа московских программистов Ассоциации групп пользователей Borland в СССР (БорАГ). Активное участие в этом процессе приняли Международный компьютерный клуб (МКК) и первый дистрибьютор Borland в СССР — СП "Интерквадро".

Одна из неформальных целей создания Ассоциации — обеспечение советских программистов и пользователей дополнительной информацией о новейших технологиях в области программных продуктов, формирование условий, позволяющих создавать конкурентоспособные программы мирового уровня с помощью инструментальных средств Borland. Деятельность Ассоциации направлена на достижение информационного паритета советских и зарубежных программистов и подразумевает создание комфортной "информационной среды" для ее членов. Ассоциация является неприбыльным, некоммерческим общественным объединением. Уникальность подхода Borland к работе с пользователями проявляется в том, что членами БорАГ могут стать не только зарегистрированные, но и "нелегальные" пользователи программных средств Borland.

Основные направления деятельности Ассоциации сближают ее с научно-техническими обществами, уже известными в СССР, как форма организации общественной активности, в их числе:

- научно-техническая пропаганда и популяризация инновационных подходов фирмы Borland к созданию и применению средств программного обеспечения, организация и проведение массовых мероприятий в сфере информатики;
- содействие распространению в СССР и за рубежом новых программных продуктов, разработанных с помощью инструментальных средств Borland;
- воспитание культуры легального использования программных средств в СССР;
- выявление и оказание содействия, включая благотворительное, перспективным, талантливым программистам и коллективам, работающим со средствами Borland;
- организация и координация деятельности Групп Пользователей Borland в СССР, взаимодействие с зарубежными Группами Пользователей.

Только в первом полугодии 1991 года БорАГ и ее членами проведено 12 научно-технических семинаров с участием представителей корпорации Borland и эк-

спертов Ассоциации (в том числе, в Москве, Ленинграде, Душанбе, Архангельске, Уфе, Казани, Севастополе).

Все члены Ассоциации бесплатно получают периодический бюллетень "БорИС" ("Borland — Инновации Сегодня"). На его страницах публикуется новейшая информация о достижениях Borland, заметки об опыте применения продукции фирмы в СССР, переводы актуальных статей из зарубежных изданий.

Для членов Ассоциации ежегодно проводится "Borland-лотерея". Корпорация Borland бесплатно предоставляет свои новейшие продукты для проведения лотереи. В рамках Лотереи-91 среди членов БорАГ разыграны 30 экземпляров наиболее популярных продуктов фирмы Borland:

- Turbo Pascal 6.0 Professional;
- Paradox 3.5;
- Quattro Pro 2.0 .

Ассоциация проводит ежегодный Всесоюзный конкурс на лучшую разработку программных продуктов, выполненную средствами Borland — "Borland-Контест". Цель конкурса: внедрение международных критериев оценки качества программных средств, популяризация разработок фирмы, поиск новых идей и оригинальных подходов к использованию программного обеспечения Borland, стимулирование и реклама разработок, которые используют продукцию Borland. В конкурсе 1990-1991 года участвовали более 300 программных разработок советских программистов со всей страны. На церемонии вручения наград, которая состоялась во время II Форума Международного компьютерного клуба (МКК) в Центре международной торговли в Москве, 41 лучшая работа была удостоена Дипломов Ассоциации групп пользователей Borland и МКК. Все дипломанты получили в качестве призов разнообразные (и новейшие!) продукты Borland, специально выделенные для этого корпорацией, а один из авторов работы, занявшей первое место, сможет совершить поездку в Калифорнию для ознакомления с работой программистов Borland.

Ассоциация групп пользователей Borland в СССР (БорАГ) приглашает и Вас стать ее членом! По вопросам о вступлении в Ассоциацию групп пользователей Borland в СССР обращаться по адресу СП "Интерквадро" (см. ниже), тел. 150-92-01 (дем. зал).

А.Зотов



Развитие программного обеспечения связано с увеличением его сложности. При этом процесс разработки программ становится очень трудоемким, а их модификация и сопровождение — затруднительными.

Введение в объектно-ориентированное программирование: язык Turbo Pascal

Первым шагом на пути преодоления постоянно растущей сложности разработки программного обеспечения стала выработка принципов структурного программирования и структурного подхода к проектированию программ. Модульность, структурный стиль и нисходящее проектирование позволили сократить сроки создания программ, упростить их отладку и сопровождение. Появились новые языки, которые прекрасно поддерживали требования структурного программирования, а старые языки модифицировались до расширений, вписывающихся в принципы структурного программирования.

Создание технологии объектно-ориентированного программирования стало вторым шагом на пути решения указанных выше проблем. Поначалу ООП, присущее только таким специализированным языкам, как Smalltalk и Simula, не получило широкого распространения в обыденной практике. Ранние разработки в области ООП были тесно связаны с исследованиями в области искусственного интеллекта и моделирования.

Поиск методов преодоления сложностей создания больших программных комплексов и их последующей модификации заставил переосмыслить принципы ООП для широкого применения. Использование этих принципов, по образному выражению одного из редакторов журнала Byte, "превратило программирование из левостороннего — логического, в правостороннее — творческое" (проводится аналогия с полушариями человеческого мозга). Желание использовать принципы ООП в широкой практике привело к появлению объектно-ориентированных расширений популярных языков программирования.

В настоящее время имеются объектно-ориентированные расширения таких языков программирования, как C, Pascal и Lisp. Для компьютеров, совместимых с IBM PC, ряд фирм выпускает компиляторы языка C++ (объектно-ориентированное расширение C), имеются компиляторы объектно-ориентированного расширения языка Pascal — Turbo Pascal фирмы Borland и Quick Pascal фирмы Microsoft. Для программирования в среде Windows фирмой The Whitewater Group разработан объектно-ориентированный язык Actor. Выпущен ряд библиотек объектов, таких как Object Professional фирмы TurboPower Software для языка Turbo Pascal, C++ Tools фирмы Rogue Wave и Zinc Interface Library фирмы Zinc Software для языка C++.

Немного истории

Впервые объектно-ориентированная техника программирования была реализована более десяти лет назад для компьютеров фирмы Apple, когда в 1980 году была выпущена объектно-ориентированная версия языка Smalltalk. В 1985 году для компьютера Lisa был

создан язык программирования Clascal. В нем был объединен синтаксис языка Pascal с понятиями классов и методов языка Smalltalk. Этот язык, совместно с библиотекой классов Lisa Toolkit, использовался для разработки программного обеспечения компьютера Lisa.

В 1984 году группа разработчиков вместе с Никлаусом Виртом, автором языка Pascal, начала работу по созданию языка Object Pascal — объектно-ориентированного расширения языка Pascal. Это расширение основывалось не на стандартном описании языка Pascal (ISO Pascal), и не на оригинальном языке Pascal (Иенсен и Вирт), а являлось расширением синтаксиса языка Apple Lisa Pascal, который, в свою очередь, унаследовал много свойств языка UCSD Pascal. Была также создана библиотека классов, названная MacApp. Первая реализация этого проекта появилась в 1985 году.

В настоящее время Object Pascal совместно с MacApp широко используется для разработки прикладных программ для компьютеров Macintosh.

ООП. Основные преимущества

В чем же состоят преимущества ООП и что лежит в их основе?

В действительности, при написании программ, даже объединяя отдельные данные в структуры и разумно разбивая программу на функциональные модули, программист неминуемо сталкивается с большим количеством мелких деталей.

ООП дает возможность отойти от мелких деталей программы (процедур и данных) и работать с ней, как с программной моделью объектов реального мира. При этом появляется возможность легкого перехода от одного уровня абстракции рассмотрения объекта (в деталях) к другому (как единое целое).

Другим преимуществом ООП является возможность использования ранее созданного программного обеспечения в процессе создания нового. При этом не требуется какой-либо переделки готового программного обеспечения и даже его перекомпиляции.

Основу ООП составляет понятие объекта. Объект (в понимании ООП), объединяя в себе декларативную и процедурную части, моделирует некоторый концептуальный или реальный объект окружающего мира. Программа, написанная в рамках ООП, представляет собой совокупность обменивающихся сообщениями объектов и моделирует некоторое реальное поведение предметов. Написание объектно-ориентированных программ облегчается тем, что программист создает программу, руководствуясь привычными понятиями той области, которую моделирует проектируемая программа.

Основа ООП — объекты

Что такое объект и какие ему присущи свойства?

Объект представляет собой совокупность данных и процедур, работающих с этими данными. Объект может моделировать как реальные явления, так и концептуальные понятия, как например, классы структур данных: список, очередь, дерево, хэш-таблица и др. В программе объекты представляют собой переменные или константы. Переменная или константа типа данных объект, называется **экземпляром** объекта. Доступ к полям данных объекта возможен только через процедуры-**методы**, определенные внутри этого объекта. Вызов процедуры, определенной внутри объекта, называется **передачей сообщения** объекту. Объединение данных и процедур, работающих с этими данными в объекте, называется **инкапсуляцией**. Инкапсуляция, наряду со строгой типизацией, ранее присущая таким языкам программирования как Ада и Модула, обеспечивает защищенность полей данных объекта от непредусмотренных модификаций, что в значительной мере повышает надежность программирования. Инкапсуляция гарантирует строго определенный, контролируемый программистом набор свойств объекта.

Другим важным свойством объектов является **наследование**. Оно заключается в том, что вновь создаваемый объект может строиться на основе существующего объекта-предка. При этом он наследует и может использовать все поля данных и методы родительского объекта. Объекты, связанные родственными отношениями, образуют иерархическую структуру (называемую **иерархией объектов**), которая может отражать иерархическую структуру моделируемого явления. Иерархическая структура позволяет легко переходить от одного уровня абстракции рассмотрения объекта (в деталях) до другого (как единое целое). Пример такой структуры показан на рисунке 1.

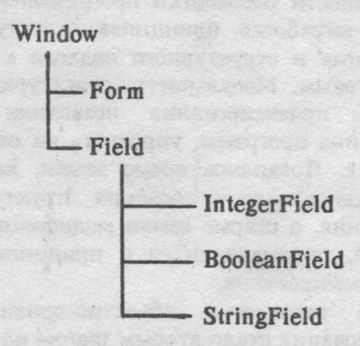


Рис.1. Иерархия объектов.

На приведенном рисунке показана иерархия объектов, состоящая из базового объекта Окно (Window), на основе которого строится объект Форма (Form). Объект Поле (Field), построенный на базе объекта Форма, является предком для трех объектов ЦелоеПоле (IntegerField), ЛогическоеПоле (BooleanField), СтроковоеПоле (StringField).

Наследование дает возможность расширять уже созданные объекты, придавая им необходимые новые свойства. Доступ к динамически создаваемым объектам одного семейства (иерархии) возможен посред-

ством совместимых по типам указателей. Поэтому, часто, для обеспечения совместимости указателей, различные объекты объединяют в одну иерархию, искусственно создавая единый для всех объект-предок, который называют **абстрактным** объектом. Абстрактный объект стоит в вершине иерархии и служит лишь для обеспечения совместимости типов указателей на объекты в иерархии. Абстрактный объект, как правило, не содержит в себе каких-либо полей данных и методов.

Объект-потомок может иметь методы с аналогичными именами, что и у объекта-предка. В этом случае говорят, что объект-потомок **переопределяет** метод объекта-предка. Переопределение дает возможность передачи одного и того же сообщения экземплярам разных объектов. Возможность использования одинаковых имен методов для различных объектов называется **полиморфизмом**.

Часто возникает необходимость использования в объекте-потомке ряда методов объекта-предка, имеющих обращения к методам, которые были переопределены у потомка. В этом случае возможно использование переопределенных методов, и такая возможность реализуется при помощи механизма виртуальных методов. Это позволяет создавать новые программы с использованием уже созданных на уровне машинного кода.

Язык Turbo Pascal фирмы Borland

Каким образом свойства ООП реализованы в объектно-ориентированном расширении языка Turbo Pascal фирмы Borland?

Для определения объекта введен новый описатель OBJECT. Он похож на описатель типа данных запись (RECORD) с одним существенным дополнением — внутри него возможно определение методов (процедур и функций), оперирующих с полями данных этого объекта:

```
Type
  {Тип данных — объект}
  ObjectName  =  Object
  {Описание различных полей данных объекта}
  DataFieldName : DateType;
  .......
  .......
  {Заголовки методов (процедур и функций),
   оперирующих над полями данных объекта}
  procedure MethodName (Parameters);
  function  MethodName : FunctionReturnType;
  .......
  end;
```

В определении объекта приводятся только заголовки методов. Процедуры и функции, которые реализуют данные методы, должны помещаться в любом месте раздела процедур или в секции IMPLEMENTATION для объектов, определенных в единице компиляции UNIT. Перед именем реализующей процедуры или функции должно стоять имя соответствующего типа объекта, отделенное разделителем (точкой):

```
Procedure ObjectName.MethodName;
Var
...
begin
...
end;
```

Определение экземпляра объекта производится в разделе переменных или констант путем указания имени экземпляра, от которого разделителем (двоеточием) отделяется имя типа объекта:

```
Var
ObjectInstance : ObjectName;
```

Доступ к полям данных объекта возможен в любом месте программы, где распространяется определение объекта, указанием имени экземпляра объекта с отделенным от него разделителем (точкой) именем поля данных. Однако при этом происходит нарушение свойства инкапсуляции, поэтому пользоваться этой возможностью не рекомендуется:

```
ObjectInstance.DataFieldName
```

В случае необходимости доступ к полям данных и методам объекта, определенным в единице компиляции UNIT, можно закрыть из других единиц компиляции (усиление свойства инкапсуляции). Для этого закрываемая группа данных и методов размещается в конце описания объекта и перед ней указывается зарезервированное слово **Private** (при этом невозможно производить расширение свойств объекта, связанных с этими полями).

Передача сообщения объекту (вызов метода) производится указанием имени экземпляра объекта с последующим указанием через разделитель (точку) имени сообщения (метода).

```
ObjectInstance.MethodName (Parameters);
```

Как и в случае работы с записями, возможно использование оператора присоединения **With** для упрощения передачи нескольких сообщений экземпляру объекта, либо для обращения к нескольким полям данных экземпляра объекта.

```
With ObjectInstance Do MethodName (Parameters);
With ObjectInstance Do ... DataFieldName ...
```

Доступ к данным объекта внутри процедур, которые реализуют тот или иной метод, осуществляется путем указания имени соответствующего поля объекта. Ссылаться на имя соответствующего экземпляра объекта или передавать его в качестве параметра не нужно, так как эта информация передается всем методам объекта через скрытый параметр **Self**. Этот параметр представляет собой не что иное, как указатель на экземпляр объекта и обеспечивает все потребности внутренней адресации. Иногда, в случае возникновения неоднозначности при обращении к переменным внутри процедур, возможно явное указание имени этого параметра перед именем поля данных соответствующего объекта:

```
Self.DataFieldName
```

То же самое относится и к вызовам других методов объекта внутри процедуры или функции, реализующей данный метод.

Если предполагается, что методы, вызываемые внутри данного метода, могут быть переопределены в объектах-потомках и необходимо осуществлять обращение к переопределенным методам, то возникают следующие проблемы. В период генерации кода компилятор не может сформировать корректные адресные ссылки на необходимый метод, так как заранее неизвестно, какая процедура будет реализовывать метод — собственная или одна из переопределенных в объектах-потомках. Процесс установки адресных ссылок в период компиляции/компоновки называют **ранним связыванием**. В случае обращения к переопределенным методам процесс установки адресных ссылок должен происходить во время выполнения программы (**позднее связывание**). Для указания компилятору того, какой тип связывания необходимо использовать, служит зарезервированное слово **Virtual**, которое записывается после заголовка метода при описании объекта:

```
procedure MethodName (Parameters); Virtual;
```

Если признак **Virtual** присутствует, то будет использоваться позднее связывание, и соответствующий метод называется **виртуальным**. В противном случае осуществляется раннее связывание и метод называется **статическим**. Если в заголовке метода появляется признак **Virtual**, то все методы в иерархии объектов (у объектов-потомков) с аналогичными именами должны иметь тот же признак и аналогичный заголовок. В случае статических методов заголовок может меняться от потомка к потомку.

Для обеспечения процесса позднего связывания в описании объекта необходимо использовать специальный метод-**конструктор**, к которому необходимо обратиться перед началом работы с каждым экземпляром объекта. Основная функция конструктора — связать конкретный экземпляр объекта с соответствующей его типу **таблицей виртуальных методов** объекта для обеспечения процесса позднего связывания. Таблица виртуальных методов содержит следующую информацию для каждого объекта: размер объекта и указатели на процедуры, реализующие виртуальные методы. Кроме того, в конструкторе возможно использование других предусмотренных программистом действий, необходимых для инициализации экземпляра объекта. Для метода-конструктора в его заголовке используется зарезервированное слово **constructor**:

```
Constructor ConstructorName (Parameters);
```

Для динамически размещаемых экземпляров объектов в процедуре **New** возможно указание вторым параметром обращение к конструктору:

```
New(PointerInstance, ConstructorName(...));
```

Также возможно использование процедуры **New** как функции:

```
PointerInstance : = New(TypeName, ConstructorName());
```

Для корректного (в случае использования механизма совместимых по типу указателей на объект) освобождения памяти, занятой динамически размещенным экземпляром объекта, возможно использование специального метода-деструктора. Помимо освобождения памяти, в деструкторе можно предусмотреть любые действия, связанные с завершением работы с объектом и освобождением памяти, занятой динамическими полями данных объекта. Для метода-деструктора в его заголовке используется зарезервированное слово **destructor**:

```
Destructor DestructorName(Parameters);
```

Компилятор автоматически подставляет в конструктор и деструктор необходимые служебные действия, поэтому, если нет необходимости использования каких-либо дополнительных действий по инициализации и деинициализации объекта, тела соответствующих процедур должны быть пустыми. В процедуре **Dispose** для экземпляра динамического объекта возможно использование деструктора:

```
Dispose(PointerInstance, DestructorName(...));
```

В дальнейшем мы более подробно опишем различия между статическими и виртуальными методами, распределение памяти под объекты, использование виртуальных методов и динамических объектов.

Д. Рогаткин
А. Федоров

Приведенный ниже словарь терминов базируется на объектно-ориентированном расширении языка Pascal — Turbo Pascal 6.0 фирмы Borland, но включает при этом распространенные термины, присущие другим языкам.

Термины объектно-ориентированного программирования

Абстрактный объект (Abstract object) — основным назначением абстрактного объекта является создание базового объекта, который затем может быть наследован другими объектами. Экземпляры абстрактного объекта никогда не создаются. Использование абстрактных объектов позволяет связать несколько объектов в одну иерархию объектов. Синоним термина в языке Smalltalk — **абстрактный суперкласс (abstract superclass)**.

Библиотека классов (Class Library) — набор готовых объектов общего назначения. Примером библиотеки классов для языков Turbo Pascal и Quick Pascal является продукт фирмы Turbo Power Software — Object Professional.

Виртуальный метод (Virtual method) — метод, адрес которого известен только в момент выполнения программы. Когда происходит вызов виртуального метода, его адрес берется из **таблицы виртуальных методов**. Это называется непрямым вызовом. В языке Turbo Pascal виртуальный метод отмечается ключевым словом **Virtual**.

Дерево классов (Class tree) — термин языка Actor, синоним термина **иерархия объектов**.

Деструктор (Destructor) — специальная процедура для высвобождения памяти, занятой объектом. Если по завершении работы с объектом (перед его удалением из памяти) необходимо выполнение каких-либо специальных действий, они должны быть выполнены внутри деструктора. Эта специальная процедура может быть вызвана в качестве параметра стандартной процедуры **Dispose** (Turbo Pascal). Деструктор объявляется использованием зарезервированного слова **destructor**:

```
destructor Done;
```

Иерархия классов (Class Hierarchy) — термин языка Smalltalk, синоним термина языка Turbo Pascal **иерархия объектов** и термина языка Actor **дерево классов**. Иерархия классов позволяет создавать классы-потомки существующих классов таким образом, что классы-потомки наследуют все данные и методы классов-предков. Иерархия классов создается объявлением абстрактного объекта в вершине иерархии и присвоением этому абстрактному объекту общего указателя на код и данные всей иерархии.

Инкапсуляция (Incapsulation) — объединение в типе данных **объект** полей данных и методов (процедур и функций), работающих с этими данными.

Класс (Class) — синоним термина языка Turbo Pascal **тип данных объект**.

Конструктор (Constructor) — специальная процедура, инициализирующая экземпляр объекта, содержащий виртуальные методы, путем установки связи между экземпляром объекта и таблицей виртуальных методов. Эта специальная процедура может быть вызвана в качестве параметра стандартной процедуры **New**. В Turbo Pascal конструктор объявляется использованием зарезервированного слова **constructor**:

```
constructor Init;
```

Метод (Method) — процедура или функция, определенная внутри объекта для работы с данными объекта. Методу доступны данные объекта без явной передачи их в качестве параметров. Возможно наследование методов. Методы могут быть статическими и виртуальными. Для виртуальных объектов существуют два специальных метода — **конструктор** и **деструктор**.

Множественное наследование (Multiple Inheritance) — наличие у объекта нескольких родителей.

Наследование (Inheritance) — процесс получения данных и методов от объекта-предка. Возможно многоуровневое наследование.

Объект (Object) — тип данных **объект**. Совокупность данных и процедур — операций над этими данными (называемых методами). Дополнительно объекты могут наследовать методы и данные объектов-предков. Термин **объект** является синонимом термина **класс (class)** языка C++.

Объекты-контейнеры (Container Objects) — объекты, способные хранить в себе другие объекты. Например: стек, очередь, дерево, динамический массив, хэш-таблица и другие подобные структуры.

Объект-потомок (Descendant object) — объект, наследующий методы и данные от объекта-предка.

```
Location = Object;        {«--- Объект-предок }
{Данные объекта и методы}
....
....
end;

Point = Object(Location); {«--- Объект-потомок}
{Данные объекта и методы}
....
....
end;
```

Объект-предок (Ancestor object) — объект, методы и данные которого наследует объект-потомок.

Параметр Self (Self Parameter) — формальный параметр, передаваемый каждому методу. Содержит указатель на конкретный экземпляр объекта.

Переопределение (Override) — процесс создания у объекта-потомка новых методов с теми же именами, что и у объекта-предка, но с новыми функциями. Это делает возможным расширение объектов. В языке Turbo Pascal 6.0 методы переопределяются прямым созданием новых методов с теми же именами. В QuickPascal ключевое слово **OVERRIDE** должно быть включено после описания переопределенного метода:

```
Type
  Point : Object(Location)
  Color : Integer;
  Procedure Show; OVERRIDE;
  Function GetColor : Integer;
  Procedure SetColor(NewColor : Integer);
End;
```

Передача сообщения (Message passing) — вызов метода для конкретного экземпляра объекта. Отличие сообщения от метода в том, что одно и то же сообщение может вызывать разные методы. Пример:

```
    Circle.Draw;
    Rectangle.Draw;
    Figures^.Draw;
```


Подкласс (Subclass) — как и термин суперкласс, используется в языке Smalltalk. Подкласс — это класс-потомок. Другими словами, если класс В наследует из класса А, то класс В является подклассом класса А.

Позднее связывание (Late binding) — ситуация, при которой адрес вызываемого метода неизвестен до момента выполнения программы. Адресация разрешается путем использования таблиц виртуальных методов с адресами методов. Также возможно раннее связывание.

Полиморфизм (Polymorphism) — возможность использования методов с одинаковыми именами для работы с различными типами данных.

Раннее связывание (Early binding) — ситуация, при которой адрес вызываемого метода известен в момент компиляции/компоновки. Также возможно позднее связывание.

Расширяемость (Extendibility) — благодаря свойству наследования, объекты могут быть расширены новыми функциями и/или полями данных. Новый метод изменяет или расширяет функции унаследованного метода и заменяет его в объекте-потомке. Возможно также добавление ряда новых методов к уже существующим.

Связывание (Binding) — процесс, в результате которого вызывающая подпрограмма получает адрес вызываемой подпрограммы. Возможно раннее и позднее связывание.

Статический метод (Static method) — метод, вызываемый с использованием раннего связывания (адрес которого известен в момент компиляции/компоновки). По умолчанию все методы Turbo Pascal являются статическими.

Суперкласс (Superclass) — в иерархии объектов каждый класс имеет только одного непосредственного предка, называемого "предком" или "суперклассом". Этот термин используется в языке Smalltalk.

Таблица виртуальных методов (Virtual method table) — таблица, хранимая в сегменте данных. Эта таблица создается для каждого объекта, имеющего виртуальные методы. В этой таблице хранятся адреса описанных в объекте виртуальных методов.

Экземпляр (Instance) — строго говоря, термин экземпляр не является специфическим для объектно-ориентированного программирования, но в связи с последним используется наиболее часто. Экземпляр — это объявленная переменная типа объект. Например, если имеется выражение Var A : Location, можно сказать, что А является экземпляром типа Location:

```
Location = Object;
{Данные объекта и методы}
....
....
end;

    Var A : Location;
```



Эта статья посвящена новому компилятору фирмы Borland — Turbo Pascal for Windows, который позволяет создавать объектно-ориентированные программы в среде Microsoft Windows.

TURBO PASCAL FOR WINDOWS

Не успели отгреметь фанфары по поводу компилятора Turbo Pascal 6.0 с прекрасной оболочкой Turbo Vision, а на улице Turbo Pascal очередной праздник: фирма Borland выпустила компилятор для создания Windows-программ — **Turbo Pascal для Windows** (TPW).

До настоящего времени для создания Windows-программ в основном использовался язык С (ряд прикладных программ создается на объектно-ориентированном языке Actor фирмы Whitewater) и большая часть разработки велась преимущественно в среде DOS с использованием ряда пакетных утилит.

TPW — первый компилятор языка программирования 3-го поколения, полностью работающий в среде Windows. В состав TPW включены все средства, необходимые для разработки Windows-программ, так что использование Microsoft SDK не требуется. Необходимо отметить, что TPW предназначен только для создания Windows-программ, для создания DOS-программ необходимо использовать компилятор Turbo Pascal 6.0. Ниже приведены основные различия между этими двумя продуктами:

TPW	Turbo Pascal 6.0
Разработка программ только для Windows	Разработка программ только для DOS
Среда Windows	Среда DOS
Объектно-ориентированная библиотека ObjectWindows	Объектно-ориентированная библиотека Turbo Vision
Только одна версия	Стандартная и "профессиональная" версии

Интегрированная среда

Интегрированная среда разработчика (ИСР), работающая в среде Windows — наиболее выразительная часть нового компилятора. Весь процесс создания

Windows-программы: ввод исходного текста, редактирование, компиляция, выполнение и отладка происходит непосредственно в среде Windows. Возможно одновременное открытие до 32 окон, которые могут быть произвольным образом расположены в рабочей области. Каждое окно может быть распахнуто или превращено в иконку. Строчное меню содержит практически те же команды, что и меню ИСР компилятора Turbo Pascal 6.0. Редактор поддерживает отмену и повтор операций.

Редактор может работать в двух режимах: режиме, совместимом с другими Windows-программами и поддерживающем набор команд стандарта SAA/CUA, и в "альтернативном" режиме, поддерживающем набор совместимых с Wordstar команд. В этом режиме возможно создание собственных комбинаций клавиш с помощью специального макрокомпилятора.

Изменение ИСР наложило свой отпечаток на ряд команд. Например, нажатие клавиш выполнения операции поиска/замены приводит к появлению блока диалога, в котором необходимо указать ряд параметров. Комбинации клавиш ^KB и ^KK позволяют выделить блок, но любое перемещение указателя отменяет выделение блока; таким образом для того, чтобы переместить блок, вместо команды ^KV теперь необходимо пользоваться областью Clipboard Windows.

Благодаря тому, что ИСР работает в среде Windows, возможно непосредственное выполнение созданной программы из среды — используя, как и в предыдущих версиях, команду Run.

Расширения языка

Похоже, стало уже традицией в каждой новой версии вносить ряд изменений в синтаксис языка. В данной версии фирма Borland предприняла еще один шаг в сторону превращения языка Pascal в язык C — была введена поддержка ASCIZ-строк. Это нововведение объясняется тем, что API Windows использует именно такие строки.

В TPW определен новый тип данных PChar как ^Char (указатель на символ) и имеется возможность присвоения этому типу данных адреса строки:

```
var
    CharPtr : PChar;
    ....
CharPtr : =  'Hello, World !';
```

Помимо этого, имеется возможность ссылки на конкретный элемент такой конструкции, например, на третий, как на CharPtr[3]. Также возможно объединение строк и удаление подстрок, определенных через указатели. Компилятор выполняет целочисленное сравнение смещений этих указателей, рассматривая их как беззнаковые слова. Компилятор не поддерживает проверки выхода за пределы диапазона (range-checking) при работе с этими строками. Такая проверка должна выполняться программистом. В состав TPW входит модуль Strings, включающий в себя функции для вычисления длины строки, копирования строк и подстрок, объединения строк, сравнения строк, преобразования строк и т.п.

С целью сокращения размера занимаемой памяти при использовании технологии объектно-ориентированного программирования в TPW введена поддержка таблиц динамических методов (ТДМ). Подробное описание этого расширения требует отдельной статьи, здесь же лишь укажем, что занимаемая таблицами виртуальных методов память существенно сокращается, особенно при использовании больших иерархий объектов.

Введен ряд новых директив компилятора:
- директива **$C** позволяет управлять атрибутами сегмента кода;
- директива **$R** предназначена для подключения ресурсов (иконок, меню, блоков диалога и т.п.) к прикладной программе;
- директива **$W** управляет генерацией пролога/эпилога, специфичного для Windows при вызове подпрограмм.

Также расширен синтаксис языка для поддержки создания и использования динамических библиотек (DLL) и динамических виртуальных методов.

Поддержка динамических библиотек

Windows позволяет прикладным программам использовать отдельные модули кода, называемые динамическими библиотеками (DLL), которые динамически компонуются к программе загрузчиком Windows. Каждая такая библиотека может содержать большое количество функций. TPW позволяет вызывать подпрограммы, расположенные в DLL и создавать собственные динамические библиотеки. Каждая подпрограмма в DLL имеет название и индекс. Вызов подпрограммы может производиться как по названию (что происходит по умолчанию), так и по индексу.

Для доступа к DLL (импорта) введена специальная директива external:

```
procedure ImportDLL; external 'DLL_Name';
```

Создание DLL-модуля заключается в описании процедур и функций библиотеки:

```
library MyDLL;
...
{реализация процедуры и функций библиотеки}
...
{exports}
...
begin
end.
```

Средство для создания ресурсов

Помимо выполняемого кода, Windows-программа может включать в себя ресурсы — иконки, меню, блоки диалога, курсоры, битовые изображения и т.п.,

для визуального создания которых в состав TPW включено средство — Whitewater Resource Toolkit (WRT). WRT является альтернативой Dialog Editor и другим утилитам для работы с ресурсами, включенным в SDK. WRT на самом деле является набором редакторов, объединенных единым интерфейсом и позволяющих редактировать блоки диалога, иконки, битовые изображения, меню, курсоры и другие ресурсы. Помимо возможности непосредственной работы с текстовыми файлами ресурсов, WRT позволяет извлекать ресурсы из файлов типа RES, EXE и DLL.

Turbo Debugger для Windows (TDW)

При необходимости отладки программ возможен вызов отладчика из ИСР. Эта возможность является новинкой, так как раньше фирма поставляла два отладчика: встроенный в среду и внешний. При вызове отладчика (команда **Debugger** меню **Run**) экран переключается в текстовый режим и появляется отладчик, внешне напоминающий отладчик, используемый в среде DOS. В TDW добавлен ряд возможностей для отладки Windows-программ: просмотр Windows-сообщений, локальной и глобальной "кучи", просмотр модулей, загруженных Windows, и отладка динамических библиотек (DLL). Обладая, с одной стороны, удобством при отладке Windows-программ, TDW не позволяет параллельно выполнять никаких Windows-программ и даже не позволяет переключать задачи клавишами Ctrl-Esc и Alt-Tab.

Разработка программ

TPW предоставляет возможность создавать Windows-программы тремя способами, в зависимости от требований к программе и опыта разработчика. При необходимости быстрой переделки программ, работавших в текстовом режиме в среде DOS, можно использовать модуль **WinCRT**. Этот модуль обрабатывает вызовы процедур **Readln**, **Writeln** и **GotoXY**. Модуль не поддерживает меню, и для того, чтобы закрыть окно по окончании работы программы (если оно не закрыто пользователем), требуется вызов специальной процедуры **WinCRTDone**. Ставший классическим пример с выводом сообщения "Hello, world!" в данном случае выглядит следующим образом:

```
program WinHello;
uses WinCrt;

Begin
   Writeln('Hello, World!');
End.
```

Отметим, что наличие только одной строки — uses WinCrt — делает программу, работающую в DOS в текстовом режиме, Windows-программой.

Для создания более профессиональных прикладных программ в состав TPW входит объектно-ориентированная библиотека классов **ObjectWindows**. Интерфейс с этой библиотекой более простой, чем с Turbo-Vision, и она также проще в изучении и использовании. Основной класс библиотеки — класс **TApplication**, наследник которого является текущей прикладной программой. Программа с использованием ObjectWindows может выглядеть следующим образом:

```
program WinHello;
uses WObjects;

type
   TDemo = object(TApplication)
    procedure InitMainWindow; virtual;
   end;

var
   WinDemo : TDemo;

procedure TDemo.InitMainWindow;
begin
   MainWindow := New(PWindow,Init(nil,'Hello!'));
end;

Begin
   WinDemo.Init('');
   WinDemo.Run;
   WinDemo.Done;
End.
```

Как видно из приведенного примера, у ObjectWindows много общего с Turbo Vision. Ряд классов и общая организация похожи, но перенос созданных программ из DOS-среды в Windows-среду требует существенной переделки.

Если же есть необходимость в написании программ, использующих вызовы Windows-функций напрямую (например, для переделки программ, написанных на языке С с использованием Microsoft SDK), в состав TPW включен модуль **WinProcs**. Полный пример программы, использующей Windows API, занимает много места. Ниже показан ее фрагмент — процедура WinMain:

```
procedure WinMain;

Var
 Window : HWnd;
 Message: TMsg;

Const
 WindowClass : TWndClass = (
   Style : 0;
   lpfnWndProc : @WindowProc;
   cbClsExtra  : 0;
   cbWndExtra  : 0;
   hInstance   : 0;
   hIcon       : 0;
   hCursor     : 0;
   hbrBackground : 0;
   lpszMenuName  : AppName;
   lpszClassName : AppName;
 )
begin
   If HPrevInst = 0 then
   begin
      WindowClass.hInstance := HInstance;
      WindowClass.hIcon :=
        LoadIcon(0,idi_Application);
      WindowClass.hCursor :=
        LoadCursor(0,idc_Arrow);
```

```
      WindowClass.hbrBackground :=
         GetStockObject(white_Brush);
      if not RegisterClass(WindowClass) then
         Halt(255);
   end;
Window := CreateWindow(
   AppName,
   'Hello, World',
   ws_OverlappedWindow,
   cw_UseDefault,
   cw_UseDefault,
   cw_UseDefault,
   0,
   0,
   HInstance,
   nil);
ShowWindow(Window, CmdShow);
UpdateWindow(Window);
while GteMessage(Message,0,0,0) do
begin
   TranslateMessage(Message);
   DispatchMessage(Message);
end;
   Halt(Message.wParam);
end;

begin
   WinMain;
end.
```

Заключение

Снижение трудоемкости разработки прикладных Windows-программ на TPW по отношению к С можно сравнить с соотношением сложности создания DOS-программ на языках высокого уровня и на ассемблере. Использование объектно-ориентированного подхода и библиотеки ObjectWindows существенно сокращает время, требуемое на разработку Windows-программ.

Можно выделить несколько категорий пользователей TPW:

- программисты, использующие Turbo Pascal и желающие создавать Windows-программы;
- программисты, ранее использовавшие Turbo Pascal и перешедшие на Microsoft C для написания Windows-программ;
- программисты, не удовлетворенные разработкой Windows-программ с использованием Microsoft SDK;
- программисты, использующие средства разработки типа Toolbook, не удовлетворенные низкой скоростью работы этих средств;
- программисты, которым необходимо перенести программы, созданные с использованием Turbo Vision, в среду Microsoft Windows;
- программисты, использующие язык PAL СУБД Paradox, желающие создавать Windows-программы, используя библиотеку Paradox Engine 2.0.

А.Федоров

Фирма IBM выпустила новую серию персональных компьютеров для мультимедиа. Эти изделия будут продаваться под маркой "Ultimedia." IBM PS/2 Ultimedia Model M57 SLC — это первый компьютер Personal System/2 (PS/2) со встроенными устройствами для мультимедиа и "самым быстрым 386 процессором". Продажа системы начнется в марте 1992 г. Стоить она будет 5995 долларов.

IBM PS/2 ActionMedia II — набор адаптеров для ввода и вывода цифрового видео- и аудиосигналов. Устройство будет продаваться по цене 1995 долларов в двух версиях — для Micro Channel Architecture (MCA) и для Industry Standard Architecture (ISA). Дополнительно можно приобрести пакет улучшения качества введенного изображения и программную систему для разработчиков за 890 и 510 долларов соответственно.

IBM PS/2 TV — устройство, которое будет состоять из видеоадаптера, тюнера, динамика и программ. Начнет продаваться за 495 долларов с 27 марта 1992 г.

IBM PS/2 TV позволяет одновременно смотреть до 70 телевизионных программ на экране PS/2, а новый адаптер F-Coupler позволяет передавать живое изображение через локальную сеть Token Ring.

IBM PS/2 TouchSelect — сенсорный экран. 12-дюймовая модель стоит 670, а 19-дюймовая — 850 долларов.

И еще IBM обещает выпустить учебный мультимедиа-диск для безработных.

Newsbytes News Network, 18 October 1991

Фирма Seagate Technology объявила о крупных финансовых убытках — 47.8 миллиона долларов — в этом квартале. Объем продаж, равный 620 миллионам долларов, остался практически на уровне предыдущего квартала. Официально убытки относят на счет реорганизации компании.

Как сообщает UPI, Дэвид Митчел, сооснователь фирмы, неожиданно подал в отставку 20 сентября после 12 лет работы в фирме.

При попытке реорганизации с фирмы было уволено 1200 человек в июле (18% персонала), и еще 450 человек в августе.

Как и фирма Conner Periferials, также объявившая об увольнениях, Seagate объясняет все свои проблемы экономическим спадом, хотя, видимо, война цен между Seagate, Conner и Quantum также сыграла свою роль.

Newsbytes News Network, 18 October 1991

При анализе прикладного программного обеспечения принято выделять электронные таблицы, текстовые редакторы, программные средства связи, системы управления базами данных и графические редакторы. Попытка отнести Object Vision к одному из этих классов вызовет некоторое затруднение, так как он является представителем интегрированного программного продукта, одной из характерных черт которого является многофункциональность.

ObjectVision: первые впечатления

Сегодня на рынке программных средств существуют и продолжают появляться пакеты полифункционального характера. К ним в первую очередь можно отнести пакеты для деловых приложений, — Form Filler, PerForm, JetForm. Они предназначены для генерации форм документов, особым образом структурированной информации (бланков, счетов, справок и т.п.), упрощения ввода и обработки информации в них, а также ведения архивов.

Такие пакеты сочетают в себе возможности текстовых, графических редакторов, электронной таблицы и рассчитаны на пользователей, не являющихся профессионалами в компьютерной области. К деловым пакетам относится и пакет Object Vision, который представляет собой нетрадиционное соединение генератора форм с WYSIWYG-интерфейсом и развитого набора математических, экономических и прочих функций.

Пакет предназначен для:
- создания и редактирования форм;
- управления формами;
- просмотра и вывода на печать форм.

Судьба любого программного продукта целиком и полностью зависит от его реальных свойств, характеристик и отличий от других пакетов идентичного функционального назначения. В чем же состоят эти отличия у Object Vision?

Во-первых, и это определяет многие свойства, Object Vision работает в среде Windows 3.0, что обеспечивает стандартный графический интерфейс и разделение ресурсов компьютера. С точки зрения пользователей к преимуществам Windows 3.0 можно отнести:
- представление информации в удобном для восприятия виде;
- большую защищенность системы от несанкционированных действий, что позволяет непрофессиональным пользователям избегать "рискованных" ситуаций;
- широкий выбор драйверов принтеров с минимальной затратой времени на их установку;
- практически неограниченный выбор палитры цветов изображения и фона;
- простота установки и настройки пакета;
- автоматический поиск файлов на дисках;
- создание пиктограмм в среде Windows.
- широкие текстовые и графические возможности, в том числе:

выбор шрифтов, их стилей и размеров;
управление расположением текста;
выбор линий различной толщины, прямоугольников с различной степенью заштрихованности;
копирование, удаление перемещение, выделение части текста или рисунка. Сложные рисунки могут быть созданы в Windows Paintbrush, а затем перенесены в Object Vision.

Вторая особенность состоит в том, что работая в Object Vision, вы можете присоединить форму к внешним данным из файлов следующих типов: ASCII, Paradox, dBASE, Btrieve, DDE (DDE — формат динамического обмена данными с другими приложениями Windows, например, Excel, Word, ImageEdit, Micrografx, Micrografx Designer, Corel Draw).

Причем после подсоединения внешних данных, дальнейшая работа с ними идет по соглашениям программы, создавшей эти файлы. Это позволяет использовать уже имеющийся опыт работы с другими прикладными программами, и ослабляет влияние чрезвычайно мощного фактора, исторического императива, которым в основном и объясняется нежелание многих пользователей изучать новое программное средство.

Мощным инструментом, отличающим Object Vision от других "деловых" пакетов, является возможность включения в форму стандартных процедур и описания процессов заполнения форм. Причем по желанию можно определить, какая часть данных будет заноситься автоматически. Иными словами, пользователь получает возможность описать логику заполнения формы, используя в качестве средства формализации лишь графическое представление процесса, что в терминологии Object Vision называется "построение дерева решений". Эта процедура может включать про-

стое множество операций, подобных сложению и вычитанию чисел и более комплексное, с проверкой различных условий перед вычислением значений и использованием логических выражений, где в качестве операндов выступают значения данных, функции или имена полей (элементов формы).

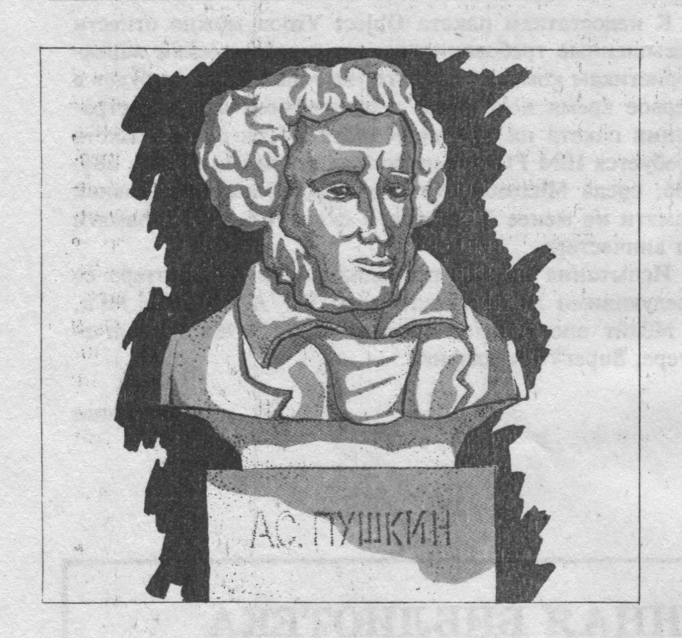
Процессу создания дерева решения обязательно предшествует процесс определения всех полей, которые будут выступать в качестве вершин дерева. Поле можно создать как средствами Object Vision, так и с помощью внешнего источника.

Область применения пакета Object Vision обширна, начиная с обычного делопроизводства, занимающего достаточно почетное место в любой области знания, и кончая, например, педагогикой.

К недостаткам пакета Object Vision можно отнести повышенные требования пакета к техническим характеристикам компьютера, что, на наш взгляд, и будет в первое время некоторым ограничением в распространении пакета на советском рынке. При работе пакета требуется IBM PC на микропроцессорах Intel 286, 386, 486, среда Microsoft Windows 3.0, объем оперативной памяти не менее 1 Мбайта и около 1.5 Мбайт памяти на винчестере.

Испытания пакета проводились на компьютере со следующими характеристиками: PC/AT 386, 25 МГц, 8 Мбайт оперативной памяти, 122 Мбайта на винчестере, SuperVGA-графика.

Л.Струкова



Поглощение Borland корпорации Ashton-Tate вызвало не менее, чем шок в рядах поклонников dBASE. Примерно такое же чувство испытали и мы. Свернет ли Borland все работы, связанные с продолжением разработки линии dBASE, или нет — пока неизвестно. Но судя по всему, в сфере баз данных происходит смена поколений, и поэтому мы спешим представить нового лидера — СУБД Paradox.

Парадоксален ли Paradox?

"И гений, парадоксов друг..."
А.С.Пушкин

Paradox — первое знакомство

Система управления базами данных Paradox является гибким современным средством построения и ведения реляционных баз данных, снабженным удобным диалоговым интерфейсом, языком программирования и средствами автоматизированного создания отчетов.

Коммерческий успех Paradox определился, начиная с версии 2.0, когда СУБД поставлялась компанией ANSA Software. Приобретение этой фирмы одним из ведущих производителей программного обеспечения, корпорацией Borland, привело к значительному расширению возможностей и улучшению пользовательских характеристик Paradox за счет применения собственных технологических новаций Borland (версии, начиная с Paradox 3.0).

В целом, по оценкам многих специалистов, Paradox приобретает черты стандарта де-факто современной информатики в области баз данных, приходя на смену семейству dBase.

Причиной этому являются следующие весомые преимущества Paradox:
- высокая производительность;
- простой и наглядный метод построения запросов по образцу (Query-by-Example);
- средства графической визуализации данных;
- ведение журнала СУБД;
- низкие требования к ресурсам компьютера (благодаря технологии VROOMM);
- наличие интерфейса со многими типами сетевых серверов баз данных (пакеты SQL Link);
- возможность (с помощью Paradox Engine 2.0) создания прикладных программ обработки баз данных на языках высокого уровня типа Microsoft C 5.1, Turbo C++, Turbo C 2.0, Turbo Pascal.

Удобство и надежность эксплуатации Paradox в локальной сети позволяет использовать его как высокопроизводительное оконечное средство доступа к удаленным данным, включая обращение к базам данных на больших ЭВМ.

По данным независимой организации Software Publishing Association, доля Paradox среди применяемых в США СУБД увеличилась за первые 5 месяцев 1990 года более, чем в два раза и достигла 20%. В мае 1990 года 40% вновь приобретаемых СУБД пришлось на Paradox — больше, чем у любого из конкурентов. К концу 1990 года Paradox захватил 34.5% рынка СУБД для IBM-совместимых компьютеров — больше любой другой СУБД в США.

Очевидно, что такие убедительные показатели неминуемо приведут ко все более широкому распространению легальных (как и пиратски похищенных) копий Paradox в нашей стране.

Общие впечатления разработчика

Сравнительные характеристики СУБД в основном совпадают в различных статьях и обзорах, посвященных базам данных. Действительно, Paradox — удобная, мощная, хорошо продуманная и изящно сконструированная система, обеспечивающая эффективность труда программиста и возможность создавать эффективно работающие приложения. Авторы надеются, что некоторые отмеченные недостатки текущих версий со временем будут устранены и Paradox будет расширен именно теми возможностями, нехватка которых кажется столь очевидной.

Главным парадоксом Paradox является, на наш взгляд, сочетание простоты и удобства с широким набором функциональных возможностей обработки данных.

Работая в диалоговом режиме меню (без программирования) можно создать довольно сложную систему обработки данных, включающую средства ввода и редактирования входных документов, поиска и обработки данных и получения разнообразных выходных документов. Диалоговый интерфейс Paradox нагляднее, чем, например, подсистема Assistant СУБД dBASE III Plus. Пользователь в каждом режиме получает доступ к крупному функциональному блоку с понятным входом и выходом, а не к элементарной операции, как в dBASE. Объяснение простое: в Paradox диалог ориентирован на удобство работы, в то время как в dBASE — на простоту преобразования в команды этой СУБД.

Достоинством диалогового монитора Paradox является активное и единообразное использование во всех режимах наряду с иерархическими меню функциональных ключей и CTRL/ALT-клавиш.

Для описания структуры базы данных достаточно заполнить простую двухколоночную таблицу, указав имена и типы полей. Дополнительное удобство — возможность использования структуры уже существующей таблицы в качестве отправного пункта при создании новой.

После описания структуры можно разработать экранные формы для ввода и редактирования данных или непосредственно начинать ввод, используя стандартные формы табличного или бланкового типа. При создании экранных форм пользователь может управлять размещением полей на экране, заданием поясняющих текстов, выбором цветовой палитры для экрана или его частей. Существует возможность связки в одной форме нескольких таблиц по составному ключу. При этом допустимы связи между записями типа "один-к-одному" или "один-ко-многим". Возможно формирование значений некоторого поля данных посредством выбора значения из домена (таблицы-справочника).

При вводе и коррекции данных могут использоваться до 16 форм ввода, связанных с текущей таблицей. Одновременно можно обрабатывать несколько форм, связанных с разными таблицами.

Язык запросов вызывает немой восторг программиста. В Paradox реализован практически полный язык запросов QBE. Синтаксис его близок к спецификациям фирмы IBM и публикациям автора языка, М.Злуфа (M.Zloof). С помощью QBE можно реализовать сложнейшие операции поиска и обработки данных. Следует отметить, что процедурные, рассчитанные на программиста средства поиска, например, в Clipper или Foxbase функционально беднее, чем декларативные (непроцедурные) средства QBE в Paradox.

Проведенные нами проверки показали, что оценки производительности поисковых средств Paradox, постоянно обеспечивающие ему одно из первых мест по скорости поиска в сравнительных обзорах СУБД для персональных компьютеров, вполне соответствуют действительности.

Отметим, что фирма-разработчик считает диалоговые возможности Paradox средствами, ориентированными на пользователя-непрограммиста. Американские программисты, конечно же, имеют в виду американских пользователей и сложившуюся в США систему, при которой в подготовку сотрудников фирмы к применению автоматизированных рабочих мест вкладываются огромные средства. В нашей стране, за исключением редких случаев, между пользователем и "голой" СУБД должен стоять программист, создающий адекватные уровню подготовки данного пользователя средства общения. Поскольку разрабатывать десятки новых приложений программированием "в лоб" неоправданно, возможности развития программ в Paradox мы будем оценивать не только с точки зрения написания конкретных приложений, но и создания инструментальных средств.

В Paradox реализован высокопроизводительный и гибкий генератор отчетов, позволяющий создавать выходные документы табличного и анкетного типа. Помимо стандартных для многих генераторов отчетов, в Paradox реализованы следующие возможности:
- вычисляемые поля отчета;
- многотабличные отчеты;
- вложенные группировки и подсчет итогов;
- получение более широких, чем каретка принтера, выходных документов (с выводом по частям).

В целом, встроенный генератор отчетов Paradox сопоставим по функциональным возможностям и простоте использования с таким специализированным пакетом, как Relational Report Writer фирмы Concentric Data Systems (последний работает с файлами dBASE).

Средства разработки сложных приложений

Для разработки приложений Paradox имеет встроенный язык программирования PAL. В сущности, PAL объединяет два языка — язык клавиатурных макрокоманд и собственно систему программирования. Любую последовательность клавиш, использованную в процессе диалога с системой, можно сохранить в файле в виде программы-макрокоманды и затем выполнить необходимое число раз. Язык программирования позво-

ляет объединять такие макрокоманды, а также включать собственные команды и функции языка. В PAL реализованы необходимые управляющие конструкции — условия, ветвления, циклы. В языке существует механизм использования процедур со средствами передачи параметров и возврата значений. Данные в языке представляются полями таблиц базы данных или переменными. Тип поля базы данных определяется при его создании и остается в программе неизменным. Тип переменной определяется типом присвоенного ей значения и может быть изменен оператором присваивания. Простая переменная может быть символьного типа, типа "дата", логического типа и одного из числовых типов — short integer (16-битное целое со знаком), currency (для денежных сумм — число с двумя десятичными знаками) и real (переменная действительного типа двойной точности). Возможно использование одномерных массивов. Компоненты массива могут быть различных типов. Переменные в процедурах могут иметь локальные или глобальные области действия.

Команды PAL обеспечивают выполнение "крупноблочных" операций Paradox — одной-двумя командами можно обеспечить просмотр и редактирование базы данных по любой из созданных экранных форм, выполнение запросов, подготовку отчетов и т.д. При этом можно использовать практически все диалоговые возможности Paradox, расширяя их средствами реакции на нестандартные клавиши или заменяя стандартные действия Paradox на необходимые в конкретной программе.

Набор встроенных функций PAL достаточно полон и включает в себя:
- функции работы с датой и временем;
- функции финансовых вычислений;
- функции получения сведений об объектах базы данных и программы, о рабочей области и параметрах системы;
- функции ввода-вывода;
- математические функции;
- статистические функции;
- функции работы со строками.

Система программирования содержит интерпретатор, редактор и отладчик. Для повышения производительности труда разработчика приложений в Paradox реализован мощный генератор приложений PProg, позволяющий существенно сократить время на проведение рутинных операций обработки данных в новых прикладных задачах. Кроме того, PProg сам по себе может служить неплохим самоучителем работе с PAL.

При использовании в локальной сети Paradox обеспечивает автоматическую блокировку, разблокирование и запрещение блокирования своих объектов в зависимости от типа операции их обработки. Все эти функции осуществляются автоматически ядром системы. При разработке сетевого приложения, например, на Clipper, функциональные режимы, которые Paradox выполняет самостоятельно, необходимо программировать вручную.

Документация и система "помощи"

Документация Paradox подробна, хорошо организована, содержит большое количество примеров. Ее удобно использовать и при начальном знакомстве с системой, и при получении справок в ходе работы. К сожалению, все это нельзя отнести к системе помощи. Несмотря на большой объем сведений и контекстную чувствительность, она обладает весьма серьезными недостатками:
- пауза при вызове Help заметна даже на машинах типа AT;
- индекс помощи практически не структурирован, поиск по нему затруднен;
- практически отсутствуют ссылки на близкие по смыслу или связанные понятия.

Кроме того, раздел со сведениями о командах и функциях языка, пожалуй, наименее полон и включает только название команды или функции и ее формат — без каких-либо комментариев. Хорошо, что этот недостаток ощущается преимущественно программистами-профессионалами, а не конечными пользователями.

Структуры данных

Paradox — "истинно" реляционная СУБД, что выгодно отличает его от семейства dBASE (Clipper, FoxBase и FoxPro, dBXL и проч.). Действительно, табличная форма хранения данных часто приводит к тому, что программные средства семейства dBASE ошибочно относят к классу реляционных СУБД. В то же время, в них отсутствует главная отличительная черта реляционных систем, языковые средства манипулирования данными, — реляционная алгебра или средства реляционного исчисления. Наш опыт показывает, что программы из семейства dBASE правильнее было бы отнести к классу СУБД с плоскими файлами, а по некоторым признакам они вообще ближе к системам файлового доступа, а не к СУБД.

Реляционность Paradox, заключающаяся в мощном, реляционно-полном языке манипулирования данными, — несомненное достоинство, обеспечивающее превосходство над семейством dBASE. В то же время, за счет великолепных технологических решений Paradox в основном лишен главного недостатка, традиционно связываемого с реляционными системами, невысокой производительности. По скорости выполнения поисковых операций он заметно опережает и Clipper, и Foxbase — чемпионов семейства dBASE.

Сама по себе реляционная модель не лишена недостатков. Основной из них — неудобство отображения семантики предметной области средствами описания схемы базы данных. Так, например, в Paradox существует понятие семейства файлов, ассоциированных с таблицей-отношением — экранные формы, отчеты, индексы и т.п. Многие операции работы с таблицей (реструктуризация, копирование, удаление) автоматически производят адекватные изменения с членами

семейства. Вызывает сожаление отсутствие таких возможностей для совокупности таблиц, входящих в некоторую базу данных. Структура многотабличной базы данных (а именно в ней в значительной мере сосредоточено семантическое описание предметной области) в Paradox не фиксируется. Связи между таблицами, ограничения целостности задаются в описаниях экранных форм и отчетов и в значительной степени в программном коде приложения, поддерживающего базу данных.

Такой способ представления семантики, очевидно, далек от идеала: средства моделирования данных коммерческих реляционных систем (в том числе и Paradox), ориентированы на описание таблиц, а не объектов реального мира, их свойств и взаимосвязей.

Справедливости ради следует отметить, что члены семейства dBASE вообще "не доросли" до того, чтобы их критиковать с точки зрения большей или меньшей семантичности моделей данных.

К недостаткам реляционной структуры данных Paradox можно отнести также отсутствие аналога текстовых полей данных с переменной длиной и возможность использования только единственного (хотя и составного) ключа таблицы для связи ее с другими таблицами.

Разработка приложений — шаг навстречу пользователю

Высокая эффективность Paradox как средства управления данными приводит к мысли о возможности его использования в составе больших разнородных систем обработки информации, где СУБД, как правило, является одной из главных компонент. За последние полтора года коллектив, к которому принадлежат авторы, выполнил следующие разработки:
- создание оболочки экспертной системы, включающей Paradox в качестве системы хранения и обработки данных;
- стыковка Paradox с системой обработки графических данных и большой программой численного анализа, реализованной на Fortran;
- разработка прикладной информационной системы на основе баз данных общим объемом более 17 тысяч строк исходного текста на PAL.

Во всех этих случаях приходилось решать проблемы взаимодействия Paradox с другими программами и пакетами программ, оптимизировать критические по времени исполнения участки PAL-программ, добиваться эффективного использования памяти.

В Paradox реализована возможность запуска программ операционной системы командами Run и RunBig. Нетрудно догадаться, что выполнение первой из них оставляет для запускаемой программы не так уж много свободной памяти, а выполнение второй требует заметного времени.

Передача параметров внешней программе возможна либо через командную строку запуска, формируемую в Paradox, либо через файл. Возврат значений — только через файл, а средства Paradox для работы с внешними файлами DOS, на наш взгляд, развиты недостаточно.

Кроме того, существует целый ряд тщательно оговариваемых в документации запретов на действия, осуществляемые внешней программой.

Таким образом, возможности связи приложений, реализованных на Paradox, с миром внешних, написанных не на PAL, программ, оставляют желать лучшего. Удачным исключением можно считать, пожалуй, простую и удобную систему экспорта/импорта данных.

Для обеспечения эффективного использования памяти при разработке программ на PAL следует:
- тщательно проектировать модульную структуру программы;
- своевременно применять в тексте программы операторы освобождения памяти, занимаемой переменными и процедурами (Release);
- избегать рекурсивных вызовов процедур;
- своевременно закрывать файлы базы данных.

При соблюдении этих несложных правил можно создавать приложения практически неограниченного размера. В то же время, если пренебречь этими требованиями, неработоспособной может стать сравнительно небольшая программа. Несмотря на это, опыт показывает, что управление памятью в больших программах на PAL может быть осуществлено значительно эффективнее, чем, например, в Clipper Summer´87 или Clipper 5.0.

В целом, система программирования PAL — это система интерпретирующего типа, поэтому, несмотря на исключительно быструю работу с базой данных, некоторые программы общего назначения в Paradox могут работать сравнительно медленно. Действительно, производительность предварительно откомпилированных программ, например, в задачах математических расчетов заметно выше. Простой пример — расчет суммы квадратов чисел от 1 до 10000. Тест проводился на машине типа PC AT с тактовой частотой 10Мгц.

Пакет	Turbo C++	Clipper	dBASE III+	PAL
Время (сек.)	3.242	19.000	256.000	69.000

В то же время, нельзя не отметить практически четырехкратное преимущество Paradox по отношению к другому интерпретатору — dBASE III Plus.

А если нужно еще быстрее?

Большие надежды разработчиков приложений Paradox связаны с новым продуктом фирмы Borland, Paradox Engine, который представляет собой библиотеку функций для работы с базами данных Paradox. С ее помощью можно выполнить из программы, написанной на языках C или Pascal, большинство операций по созданию и ведению баз данных, провести поиск в таблицах, разработать сетевые приложения, использу-

ющие богатые средства Paradox для коллективного доступа к базам данных. К сожалению, в текущей версии Paradox Engine нельзя непосредственно из программы на языке высокого уровня использовать такие "изюминки" Paradox, как экранные формы, запросы и отчеты.

Разработчики фирмы Borland в рамках Paradox Engine 2.0 реализовали не только возможность работы с файлами Paradox из языка Turbo Pascal, но и программный интерфейс к DLL (динамически компонуемым библиотекам) Microsoft Windows 3.0. В результате программисты всего мира получили уникальный набор инструментальных средств для работы с базами данных, до сих пор отсутствовавший в среде MS-DOS.

Диалог должен быть современным!

Набор команд PAL, используемых для организации диалога с пользователем, существенно превосходит возможности dBASE, но уступает аналогичным средствам такой системы программирования, как Clipper. Реализация меню в Paradox осуществляется в так называемом Lotus-подобном стиле (горизонтальное разворачивающееся меню со строкой-комментарием). В то же время, с помощью PAL и процедур из поставляемого с Paradox набора ToolKit (полезные примеры программ на PAL) можно запрограммировать и более удобные всплывающие меню.

В языке программирования PAL, к сожалению, отсутствует понятие окна и средств для работы с окнами. Вместо этого введено понятие занавеса, экранной поверхности, на которой Paradox производит свои действия. Эта поверхность может быть закрыта занавесом (возможно, прозрачным), на котором отображается экранный вывод PAL-программы. При программировании это не всегда удобно, да и выглядит как-то несовременно. Не поддерживается интерфейс с мышью. Оставляет желать лучшего и управление звуком.

Команды ввода-вывода на экран не позволяют сохранять, очищать или закрашивать область экрана (занавеса). Ввод данных в базу удобно реализуется в виде экранных форм, а вот средства для запроса каких-либо параметров, не являющихся полями базы данных, недостаточны. Нельзя, например, менять форму и цвет курсора. В целом, все, что касается программно управляемых средства диалога, оставляет широкое поле для совершенствований.

Следует отметить, что в ближайших версиях и сам Paradox, а не только Paradox Engine, будет работать в среде Windows. Первые шаги в этом направлении сделаны в версии 3.5. В ее дистрибутив входят .PIF файлы и иконки Windows для Paradox и его генератора приложений PProg. На рубеже 1992 года ожидается и появление специализированной версии Paradox для Windows.

В заключение следует отметить, что предъявление программистами таких требований к Paradox, который является проблемно-ориентированной программой для ведения баз данных (СУБД), свидетельствует, в первую очередь, о его огромном потенциале в качестве средства создания автоматизированных рабочих мест различного назначения. Действительно, вряд ли кто-нибудь будет ожидать многого от программной системы, которая не удовлетворяет пользовательским требованиям сама по себе. В то же время, мощную, гибкую и удобную для разработчиков прикладных программ систему всегда хочется улучшить одной-двумя ценными характеристиками.

Д.Хан-Магометов, К.Раннев, А.Зотов

Zenith Data Systems демонстрировал на выставке Comdex в Лас-Вегасе новый компьютер-блокнот (notebook) MastersPort 386SLe. ZDS выпустила также настольный компьютер на процессоре 486SX.

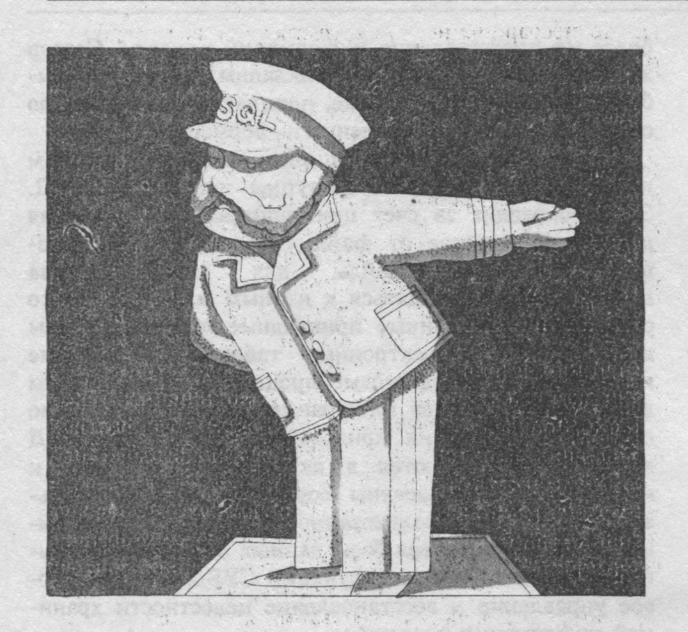
MastersPort 386SLe выполнен на новом 25-мегагерцовом процессоре i80386SL, который работает на 25 процентов быстрее, чем ранее имевшийся 386SX. Компьютер имеет 85-мегабайтный жесткий диск, улучшенный параллельный порт, дисплей VGA, и 2 Мбайта ОЗУ. Машина весит 7 фунтов (3.15 кг) и начнет продаваться в США в ноябре за 5,000 долларов.

Примененный в рабочей станции Z-486SX/25E процессор 80486SX имеет производительность на уровне 486DX, работающего с такой же тактовой частотой. Разница состоит в том, что в нем не встроен модуль вычислений с плавающей точкой.

Компьютер имеет графический адаптер Z-649 Texas Instruments Graphics Architecture (сокращенно — TIGA), обеспечивающий разрешающую способность 1024x768 пикселей и до 256 цветов.

Z-486SX/25E с винчестером емкостью 200 Мбайт называется Model 200 (6199 долларов), а без винчестера и видеоадаптера — Model 1 (4049 долларов). Обе машины имеют 4 Мбайта ОЗУ с возможностью расширения до 32 Мбайт на основной плате, три слота EISA (Extended Industry Standard Architecture), и поддержку математического сопроцессора Weitek 4167 .

Newsbytes News Network, 28 October 1991



Массовое освоение пользователями техники персональных вычислений быстрыми темпами формирует потребности совместного использования информации. Эта тенденция ведет к созданию баз данных коллективного пользования, в которых центральным технологическим звеном становятся серверы баз данных.

Серверы баз данных

Введение

Программные средства серверов баз данных обеспечивают реализацию многопользовательских приложений, централизованное хранение, целостность и безопасность данных. Производительность серверов баз данных на порядок выше по сравнению с традиционными файл-серверами, которые используются в локальных сетях. Кроме того, SQL-серверы баз данных для ПЭВМ обеспечивают многие возможности больших и мини-ЭВМ по ценам значительно более низким, соответствующим ценам ПЭВМ.

Данный обзор посвящен анализу основных проблем в развитии технологии серверов баз данных, а также рассмотрению возможностей некоторых зарубежных программных средств, реализующих новую технологию.

Прежде чем перейти к такому анализу, хотелось бы более подробно рассмотреть основные принципы работы и отличия серверов баз данных от традиционных файл-серверов.

Локальные вычислительные сети (ЛВС), первоначально созданные для совместного использования дорогостоящего периферийного оборудования, эволюционировали со временем до такой степени, что стал возможен доступ многих пользователей к одним и тем же файлам, и для многих персональных систем управления базами данных были предложены сетевые версии. Однако скоро стало ясно, что сетевые СУБД, основанные на модели файл-сервера, недостаточно мощны для мини-ЭВМ. Но, почему? Ответ на этот вопрос — основа для понимания принципов работы серверов баз данных.

Файл-серверу на микропроцессоре 386 с тактовой частотой 25 МГц и объемом памяти жесткого диска 40 Мб трудно соперничать с мини-ЭВМ. Конечно, можно использовать устройства оптической памяти, чтобы имитировать огромную память миникомпьютера. Но в нагруженной сети неизбежно встанет проблема производительности, а также безопасности и целостности данных.

Производительность является проблемой не потому, что современным 386 процессорам не хватает требуемой мощности, а потому, что сегодняшние файл-серверы используют принцип "все или ничего" для исполнения запросов рабочих станций. Полные копии файлов базы данных постоянно перемещаются вперед-назад по сети. Проблемы с безопасностью и целостностью данных возникают из-за того, что файл-серверы изначально не были сконструированы с учетом целостности данных и их восстановления в случае аварии, неявного распараллеливания задач (implicit concur-

rency) и централизованного контроля управления данными, типичными функциями, выполняемыми СУБД на средних и больших ЭВМ.

Архитектура "клиент-сервер" (CSA — Client Server Architecture) заменила модель "файл-сервера" на более мощную, состоящую из "клиентов" и "серверов", что позволило совместить достоинства однопользовательских (высокий уровень диалоговой поддержки, дружественный интерфейс и низкую цену) и более крупных компьютерных систем (поддержка целостности и защита данных, многозадачность).

В классическом понимании СУБД представляет собой набор программ, позволяющий создавать и поддерживать базу данных. С функциональной же точки зрения СУБД выглядит иначе.

Функционально СУБД состоит из трех частей: ядра базы данных (engine), языка (language) и инструментальных средств программирования (tools). Инструментальные средства относятся к интерфейсу пользователя, или внешнему интерфейсу, который мы используем. Они могут включать процессор обработки запросов на языке SQL или QBE (Query by Example — язык запросов по образцу). Язык — это совокупность процедурных и непроцедурных команд, поддерживаемых СУБД. Ядро базы данных выполняет все остальные функции, которые обычно включаются в понятие "обработка базы данных". Термины "ядро" (engine), "сервер" (server) и "внутренний интерфейс" (back end) являются синонимами. Основная идея модели "клиент-сервер" заключается в том, чтобы располагать серверы на мощных машинах, а приложениям, использующим языковые компоненты СУБД, обеспечить доступ к ним с менее мощных машин-клиентов посредством внешних интерфейсов. Тем самым использование серверов базы данных позволяет задействовать ресурсы как самого сервера базы данных, так и клиентов, менее мощных компьютеров. Обособляя операции работы с базой данных, архитектура "клиент-сервер" позволяет объединить усилия всех компонентов системы. Сервер базы данных или внутренний интерфейс (back-end) обслуживает базу и отвечает за целостность и сохранность данных, а также обеспечивает операции ввода/вывода при доступе клиента к информации.

Ввод/вывод в базе данных основан не на физическом дроблении данных, а на логическом. Это означает, что вместо того, чтобы отправлять клиентам полные копии файлов базы данных, сервер посылает им только логически необходимые порции, уменьшая тем самым трафик сети.

Вообще говоря, серверы баз данных совсем не обязательно используют язык SQL. Но большинство серверов, разрабатываемых сегодня, делает это, т.к. SQL очень удобен как язык для описания логических подмножеств базы данных.

В архитектуре "клиент-сервер" интерфейс пользователя, отображения и запросы хранятся отдельно от системы управления реальными данными и их фильтров. Внешние прикладные программы сориентированы более на представление информации клиенту. Сервер же обрабатывает запросы прикладных программ, выбирает необходимые данные, посылает их клиентам по сети и производит обновление информации.

Серверы баз данных расширяют диапазон программ пользователей, которым доступны данные в СУБД. Это происходит за счет централизованного хранения данных (в отличие от файл-серверов, поддерживающих отдельные файлы для каждого типа приложений). Обращаться к данным могут не только специально написанные прикладные программы базы данных, но и электронные таблицы, настольные издательства или текстовые процессоры. Сервер базы данных обеспечивает интеграцию данных независимо от использующих их приложений. Данные в СУБД всегда поддерживаются в актуальном состоянии и могут быть использованы совместно многими пользователями. Централизованное хранение и программные средства сервера базы данных обеспечивают выполнение таких важных функций СУБД, как диалоговое управление и восстановление целостности хранимой информации после сбоя.

Все перечисленные выше функции серверов баз данных будут более подробно рассмотрены в следующих разделах.

Тенденции развития рынка

Рынок программных средств серверов баз данных в настоящее время весьма активно развивается и делится на две части. Одна из них — это крайне необходимые приложения, позволяющие пользователям и разработчикам на основе ПЭВМ и серверов баз данных создавать системы, для которых ранее использовались универсальные или мини-ЭВМ. Это рынок, где развивались и впредь будут развиваться административные информационные системы, выполненные на Си, Коболе или программных средствах четвертого поколения (SQL Windows или dBase). В таких системах для разработчиков и пользователей очень важны возможности создания надежных программ и обеспечения их работоспособности в многопользовательском режиме в локальных сетях ПЭВМ.

Другая часть рынка программных средств серверов баз данных — это приложения для создания структур автоматизации типа "групповое обеспечение/офис" (groupware/office). Это системы, в которых ядро многопользовательской базы данных используется для автоматизации работы офиса, будь то электронная почта или электронные таблицы, связанные с базами данных (пакеты Lotus 1-2-3, Excel).

В настоящее время на рынке зарубежных ПС серверов баз данных имеется около десяти основных продуктов [2]. Сравнительные характеристики этих продуктов приведены в таблице. Среди наиболее популярных программных продуктов можно назвать SQL Server (Microsoft), SQL Base Server (Gupta Technology), Oracle Server (Oracle Corp.).

Таблица сравнительных характеристик серверов

Продукт/Версия/ Компания	Возможности ядра базы данных: Объем оперативной памяти	Объем внешней памяти	Макс. число: открытых окон	Макс. число: полей записи	Макс. длина: текстового поля	Макс. длина: числового поля	Типы полей
	1	2	3	4	5	6	7
Empress RDBMS 4 Empress Software	1 МБ	15 МБ	HD	9999		15	I, C, F, D, M, Bi,U
NetWare SQL 2.10 Novell	1 МБ	5.4 МБ	255	HD	255	15	I, C, L, F, D, M
Oracle Server 1.0 Oracle	8 МБ	20 МБ	HD	254	64	38	I, C, N, D
SQL Server 1.1 Microsoft	8 МБ	14 МБ	10.000	255	HD	15	I, C, F, P, L, D, M, Bi, U
SQL Server 4.0 Sybase	6-8 МБ	30 МБ	HD	250	255	HD	I, C, F, L, D, U
SQLBase 4.0.1 Gupta Technologies	4 МБ	1 МБ	HD	254	HD	22	I, C, F, D, P, D, Bi
VIA/DRE 1.2.3, VIA Information Systems	640 КБ	4.5 МБ	HD	HD	HD	16	I, C, F, L, D, Bi
XDB 2.30 XDB Systems	640 КБ	3.5 МБ	246	400	4056	15	I, C, N, D, Bi

HD - в зависимости от типа компьютера

В графе 7 приняты следующие сокращения типов полей:

I (Integer) - целое;
C (Character) - символ;
F (Float) - число с плавающей точкой;
P (Packed) - упакованный формат;
L (Logical) - логический;
D (Date) - дата;
M (Memo) - текстовое поле;
Bi (Biob) - двоичный формат;
U (User Defined) - определяемый пользователем.

Таблица сравнительных характеристик серверов (продолжение)

Продукт/Версия/ Компания	SQL, реляционные возможности: Кол-во правил Кодда, поддерживаемых СУБД	Реляционные возможности	Реализация интерфейса 3-го покол. (3 SQL)	Язык программирования
	8	9	10	11
Empress RDBMS 4 Empress Software	12	V PK FK NQ NV RI	B	C
NetWare SQL 2.10 Novell	7	V OJ NQ	B	Cobol, C, Assembler Basic, Pascal
Oracle Server 1.0 Oracle	10	V Sn Do PK FK OJ NQ NV	B	Fortran, Cobol, C
SQL Server 1.1 Microsoft	10	V Sn Do PK NQ NV RI FK OJ	C	C, Cobol, Basic
SQL Server 4.0 Sybase	10	V Do PK FK OJ NQ NV RI	C	C
SQLBase 4.0.1 Gupta Technologies	10	V Sn PK OJ NQ NV	B	Fortran, Cobol, C Assembler, Basic
VIA/DRE 1.2.3, VIA Information Systems	8	PK FK NQ NV	B	C, Cobol, VIA
XDB 2.30 XDB Systems	12	V Sn Do PK FK RI OJ NQ NV	B	Fortran, Cobol, C, Assembler, Pascal

1. В графе 9 приняты следующие сокращения:
V (Views) - представления пользователя;
Sn (Snapshots) - выборки;
D (Domains) - домены (области определения столбцов отношений);
PK (Primary Keys) - первичные ключи;
NQ (Nested Queries) - рекурсивные запросы;
NV (Null Values) - неопределенные величины;
RI (Referential Integrity) - целостность ссылок.

2. В графе 10 приняты следующие сокращения :
E (Embedded SQL) - встроенный SQL;
C (Call Interface) - интерфейс вызова;
B (Both) - обе возможности.

Таблица сравнительных характеристик серверов (продолжение)

Продукт/Версия/ Компания	Внешний интерфейс поддерживаемый данным сервером БД	Сетевая поддержка: сетевые операционные системы	сетевые протоколы
	12	13	14
Empress RDBMS 4 Empress Software	Empress 4GL, Empress Report Writer, Interactive SQL	DEC AR NFS RFS	TCP/IP
NetWare SQL 2.10 Novell	SQL File,RaSQL,XQL, Xtrieve Plus, R&R, Quicksilver, dBXL, Alpha 4, Lotus	N N386	IPX/SPX
Oracle Server 1.0 Oracle	Pro*C,SQL*Forms, SQL*Reportwriter, SQL*Menu,SQL*Plus, Oracle,Quicksilver/ dBXL for Oracle, JYACC, JAM, others	N N386 LM LS B DEC AS	TCP/IP, Named Pipes NetBIOS
SQL Server 1.1 Microsoft	DataEase,Paradox 3.5, Advanced Revelation, mdbs Object/1,Vinzant BASIC,Datawiz DBSQL, others	N N386 LM LS NP DEC	TCP/IP, Named Pipes
SQL Server 4.0 Sybase	Lotus 123,Microsoft Excel, APT Workbench, JYACC, JAM	N N386 LM LS DEC	Named Pipes
SQLBase 4.0.1 Gupta Technologies	SQLWindows,ZIM,dBXL, Quicksilver,Clipper, JAM/DBi,NOMAD,SQL Commander,SQL	N N386 LM LS B S 3S 3+ MS	TCP/IP, Named Pipes NetBIOS
VIA/DRE 1.2.3, VIA Information Systems	VIA/COOL,NPL/R,XI+, Caseworks-PM,C language tools	N N386 LM LS B NP T	TCP/IP, Named Pipes, NetBIOS
XDB 2.30 XDB Systems	R&R Report Writer, SQR,JAM,XDB-SQL, Freeform,XDB-Graph	N N386 NP	NetBIOS, IPX/SX

В графе 13 приняты следующие сокращения:

N - Net Ware;	B - Banyan;	AT - Apple Talk;
N386 - Net Ware 386;	DEC - DECNet;	T - TOPS;
NtB - Net BIOS;	3S - 3Share;	AR - Apollo Ring;
LM - LAN Manager;	3+ - 3 Plus Open;	NFS - NetWork File System;
LS - IBM LAN Server;	SL - StarLAN;	MS - MS-Net.
	AS - Appleshare;	

По мнению зарубежных экспертов [3] SQL-серверы баз данных являются новейшим и потенциально самым мощным приложением для сетевой обработки данных. На сегодня рынок программных продуктов серверов баз данных не очень велик, но по прогнозным оценкам специалистов в ближайшие пять лет количество реализованных пакетов может возрасти в 45 раз, а прибыли от их реализации — более чем в 100 раз.

Операционная среда сервера

Под операционной средой сервера базы данных мы понимаем возможности операционных систем компьютеров и сетевых ОС. Каждый сервер базы данных может работать на определенных типах компьютеров и сетей (все типы сетевых ОС и протоколов, поддерживаемых конкретным сервером базы данных, приведены в таблице сравнительных характеристик серверов. Операционными системами серверов могут быть DOS, OS/2, Xenix, Unix а также DEC VMS. Рабочие станции пользователей обычно работают под управлением DOS, OS/2, Xenix или Unix. Существуют также возможности смешанного использования различных операционных систем. Выбор наиболее подходящей операционной системы необходимо делать с учетом многих факторов, включая стандарты, используемые в организации, предполагаемые объемы данных и вычислений, принимая во внимание перспективы совершенствования и расширения системы.

Выбор локальной вычислительной сети для сервера базы данных производится также по комплексным критериям. Серверы баз данных SQLBase (Gupta Technology), XDB (Software XDB DBMS) и VIA/DRE (Information Systems) работают под управлением DOS и OS/2 и поддерживают сетевой протокол NetBIOS. Сервер XDB работает кроме того под управлением некоторых типов ОС Unix. При установке на сервере операционной системы IBM OS/2 Extended Edition рабочие станции клиентов могут использовать как DOS, так и OS/2. Эта ОС интегрируется с сетевым сервером фирмы IBM и использует протокол OS/2 Extended Edition Communication Manager. Сервер SQL Server (Microsoft/Sybase) требует OS /2 для сервера и поддерживает рабочие станции клиентов на базе DOS и OS/2. Ему требуется поддержка Named Pipes и других сетевых функций управления прикладного программного интерфейса (LAN Manager API). Поскольку SQL Server является одновременно и продуктом фирмы Sybase, его пользователи могут переходить к Unix или VMS, которые фирма Sybase поддерживает без изменения прикладных кодов.

Сервер Netware SQL (Novell) работает как VAP (Value Added Process), под управлением Novell NetWare или NLP (Network Loadable Module — сетевой загрузочный модуль), в сети NetWare 386 и использует систему управления транзакциями TTS (Transaction Tracking System).

Сервер базы данных фирмы Oracle (Oracle Server) поддерживает NETBIOS, IPS/SPX и Banyan Vines.

В целом, при разработке систем прежде всего необходимо оценить будущие прикладные задачи системы, а затем искать оптимальную операционную систему и сетевую конфигурацию.

Использование языка SQL

Большинство серверов баз данных поддерживает полный набор команд языка SQL, который соответствует либо стандарту ANSI, либо стандарту IBM. Кроме того, они обычно включают определенные расширения стандарта SQL. Например, сервер XDB совместим с SQL СУБД DB2 фирмы IBM. Сервер XDB очень близко подходит к архитектуре вложенного языка SQL фирмы IBM.

Ряд программных продуктов поддерживают специальные типы данных и функций. SQL-продукты, реализующие ядро СУБД в Ingres, Empress и Interbase, позволяют разработчикам определять собственные функции, используемые вместе с SQL-командами подобно любым другим SQL-функциям. Интеллектуальная база данных Ingres (версия 6.3) имеет средства, позволяющие определять новые типы данных и операторов. СУБД XDB и Oracle обрабатывают рекурсивные запросы.

Большая часть SQL-серверов (ядро) может хранить описания базы данных в системном каталоге. Этот каталог обычно бывает доступен пользователю — для обращения к нему используются те же SQL-запросы. Информация, хранящаяся в каталоге имеет чрезвычайно важное значение. Обычно там хранятся описания таблиц, колонок, индексов колонок и имена физических файлов. Некоторые каталоги хранят статистическую информацию о таблицах — число строк, наибольшие и наименьшие значения в колонках. Системный каталог поддерживает данные, которые администратор может использовать при восстановлении системы.

Реляционные СУБД могут также использовать эту информацию для оптимизации SQL-запросов. Однако эти данные поддерживаются не всеми СУБД. Примером может послужить, в частности, Oracle — одна из реляционных СУБД, не имеющих средств хранения в каталоге статистической информации.

Управление транзакциями

СУБД должна поддерживать целостность базы данных при выполнении транзакций. SQL-серверы особенно мощны в реализации этого требования. Транзакция — это набор команд работы с базой данных, который выполняется каждый раз при решении определенной задачи. Обычно при выполнении транзакции обновляется несколько таблиц и индексов, связанных с этими таблицами.

Для того, чтобы гарантировать синхронизацию обновления и целостность данных, в серверах обычно используют принцип "все или ничего", то есть либо вносят все обновления транзакции в базу данных, либо не вносят ни одного из них. Это происходит следующим образом. Обработка транзакции начинается после получения команды "начало транзакции". Некоторые системы используют специальную команду, другие воспринимают в качестве неявной команды начало программы. Завершает транзакцию команда SQL Commit, она и служит сигналом серверу базы данных о том, что все полученные изменения могут быть внесены в базу данных. Если же в процессе обработки транзакции произошел сбой, то все изменения транзакции автоматически игнорируются, а база данных сохраняется в неизменном состоянии.

Управление транзакциями включает также такие важные элементы как автоматическая блокировка данных на время внесения изменений и определение состояний взаимоблокировки. Блокировка (Locking) предотвращает одновременное обновление данных несколькими пользователями. Некоторые серверы требуют явного указания операторов блокировки в тексте транзакции, но автоматическая блокировка предпочтительнее. Блокировать (в зависимости от типа системы) можно отдельные записи, страницы или только всю таблицу. Страницы представляют собой физические блоки, размеры которых варьируются от 1 до 4 Кбайт и включают несколько записей. Блокировка на уровне записи обеспечивает наивысший уровень параллелизма. Блокировка на уровне таблицы позволяет в каждый момент времени обновлять таблицу только одному пользователю. Блокировка на уровне страницы находится где-то посередине. Большинство серверов реализуют блокировку на уровне записи или страницы. Единственной реляционной СУБД, которая предоставляет возможности для динамического выбора уровня блокировки между уровнем записи и уровнем таблицы является СУБД Informix-Online.

Серверы базы данных должны автоматически определять состояние взаимоблокировки (deadlock). Это состояние возникает тогда, когда программа А читает запись, которая заблокирована программой В, а программа В читает другую запись, которая заблокирована программой А. Каждая из программ в этом случае будет ожидать завершения другой. Следовательно, обе программы окажутся в состоянии взаимного ожидания, которое и называется взаимоблокировкой. Сервер базы данных должен выявлять подобные состояния и автоматически прерывать выполнение одной из транзакций, выводя тем самым другую из вечного ожидания. Прерванная транзакция после исключения возможности ее блокировки будет выполнена сначала. В некоторых системах отработка подобных ситуаций выполняется автоматически, в других требуется определенный опыт программирования, поскольку при определении состояния взаимной блокировки программы будут выполнены повторно сначала.

Восстановление базы данных

Возможности восстановления базы данных являются критическим фактором в выборе сервера. При возникновении сбоя в системе или порчи информации на диске данные, не записанные физически в резервной копии базы данных, могут быть восстановлены из системных буферов. А чтобы гарантировать, что ни одно из этих данных не пропадет, сервер базы данных должен поддерживать специальный текущий файл регистрации всех прошедших транзакций.

В случае выхода из строя системы или жесткого диска, обычная процедура состоит в том, чтобы переписать базу данных с резервной копии. Но в таком случае теряются все обновления, которые прошли с момента создания резервного файла вплоть до выхода из строя жесткого диска. Сервер базы данных должен поэтому иметь специальные утилиты, позволяющие считывать текущий файл регистрации транзакций и восстанавливать их. Такие процедуры восстановления носят название "Roll Forward". Сервер IBM Database Manager OS/2 Extended Edition на сегодня единственный из пакетов, не имеющий подобной процедуры.

Некоторые серверы баз данных имеют так называемый "терпимый к сбоям" режим работы (fault-tolerant processing), сущность которого состоит в следующем. Две системы ведут одновременно две копии одних и тех же данных. Если одна из систем выходит из строя, то другая продолжает работать, сохраняя целостность базы данных. Метод крайне дорогостоящий, но в некоторых приложениях (работающих в реальном времени) без подобных издержек просто не обойтись.

Оба названных метода поддерживают программные продукты Ingress, SQL Server, SQL Base и Informix-Online.

Целостность данных

СУБД должны обеспечивать выполнение правил целостности или взаимной согласованности данных, включая целостность объектов, ссылок и реализацию правил, вводимых пользователем.

Целостность объектов предполагает уникальность каждой записи в таблице. Чтобы обеспечить целостность объектов, в таблице выделяется поле (или группа полей), значение которого всегда должно быть определено, значение этого поля (полей) однозначно определяет запись и называется ключом.

Целостность ссылок гарантирует согласованность значений полей в нескольких реляционных таблицах. Непосредственно (с помощью оператора языка SQL CREATE TABLE наподобие используемого в DB2) поддерживает целостность ссылок сервер фирмы IBM Database Manager OS/2 Extended Edition. Этот вариант наиболее предпочтителен. Сервер XDB фирмы XDB Systems позволяет пользователям определять взаимосвязь ссылок с помощью команды языка SQL ALTERTABLE. В SQL Server фирмы Sybase и СУБД

Ingres используются т.н. триггеры. Триггеры представляют собой процедуры, которые автоматически выполняются перед каждым обновлением базы данных. Администратор базы данных может выполнять триггеры на сервере базы данных.

Взаимодействие серверов

На больших предприятиях может возникнуть необходимость в организации взаимодействия между несколькими серверами или их совместной работы с большими и мини-ЭВМ. Соединяя системы вместе, следует выбрать одну из альтернатив: либо организовать простейший вариант обмена данными между компьютерами и выбрать способ загрузки данных в главную ЭВМ, либо пытаться эмулировать работу распределенной базы данных. С точки зрения пользователя в распределенных базах данных несколько серверов работают "прозрачно", т.е. как один очень большой сервер. У администратора же базы данных появляется возможность более равномерно распределять загрузку мощностей, в то время как вся информация остается доступной пользователям, работающим на нескольких рабочих станциях. Некоторые СУБД позволяют выполнять только поиск информации в таких системах, другие поддерживают полное распределенное обновление. Примером реляционной СУБД, полностью поддерживающей распределенную базу данных, может служить сервер VIA/DRE.

Другой аспект взаимодействия — это взаимодействие программного обеспечения, обеспечивающее программам доступ к удаленным базам данных на основе модели peer-to-peer. Такие программы обычно используют специальный протокол APPC (Advanced-programm-to-programm-communication) фирмы IBM. Используя APPC, программы могут выполнять большую часть своей работы в локальной сети, сохраняя средства доступа к ограниченным наборам данных на больших или мини-ЭВМ. Этот сценарий обычно реализуется, когда сетевые прикладные программы нуждаются в обновлении или поиске записей, хранящихся в базе данных большой ЭВМ. Протокол APPC используется, например, в пакете SQLNet, который фирма Gupta Technology разработала для обеспечения совместимости своего сервера SQL Base с СУБД DB2 фирмы IBM. Microsoft также с помощью APPC недавно внедрила "связку" для своего сервера базы данных SQL Server и СУБД DB2.

Управление ресурсами

У серверов баз данных очень широк диапазон ресурсов, которые они потребляют, и требований по их управлению. Программы серверов баз данных, ориентированных на большие и мини-ЭВМ, не соответствуют продуктам для ПЭВМ. Серверы, адаптированные с больших ЭВМ, могут иметь также поддержку файловых форматов для ПЭВМ и адекватные возможности импорта/экспорта файлов ПЭВМ. Ограничения по требованиям к памяти на каждого пользователя у серверов могут варьироваться от 30 до 500 Кбайт, при наличии внешних систем требования могут быть еще более широкими.

Серверы баз данных используют разные форматы хранения данных: некоторые из них хранят записи произвольной длины, другие — только фиксированной длины. Это ведет к большим различиям в требованиях к дисковой памяти. Одни серверы автоматически могут вновь использовать пространство памяти, занятое удаленными записями, другие могут требовать для этого выполнения специальных утилит.

Наиболее совершенные серверы имеют специальные утилиты настройки и мониторинга базы данных, позволяющие оптимизировать производительность работы.

Серверы могут также поддерживать команды языка SQL GRANT и REVOKE, которые обеспечивают безопасность базы данных.

Для эффективной работы с серверами баз данных, в силу их достаточно высокой сложности, требуются администраторы, хорошо ориентирующиеся в вопросах управления данными. Поэтому вопросы обучения и подготовки персонала играют здесь далеко не последнюю роль.

Поддержка внешних прикладных программ

Серверы баз данных рассчитаны на поддержку большого количества различных типов приложений, и разработчики нуждаются в инструментальных средствах для их использования. Поэтому следует особо остановиться на прикладных разработках и инструментальных средствах, появляющихся на рынке вместе с серверами.

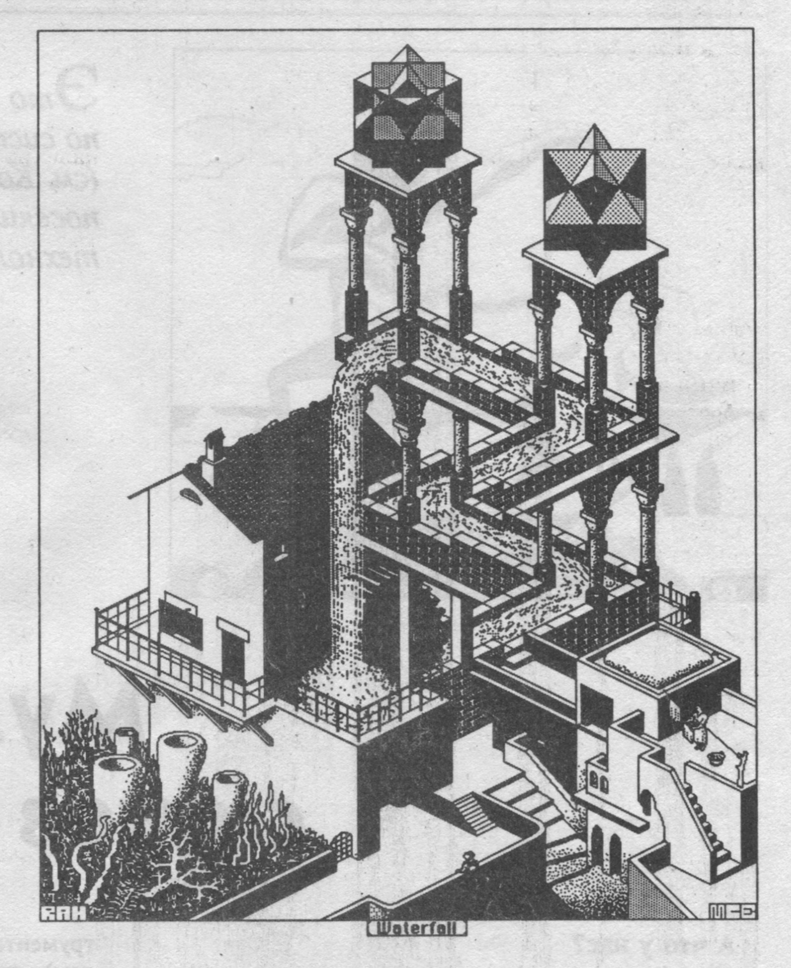
В зависимости от своих прикладных задач, пользователи могут нуждаться в различных типах инструментальных средств для реализации интерфейса с базой данных: объектно-ориентированные средства, электронные таблицы, текстовые процессоры, графические пакеты, настольные издательства и другие. Пользователю могут быть также необходимы специальные инструментальные средства, которые нужны при разработке сложных программ, требующих применения экспертных систем или других программ искусственного интеллекта.

Организация пользователя может уже иметь большой опыт работы с однопользовательскими СУБД, такими как dBASE, DataEase, Paradox, Revelation или совместимыми с dBASE, так называемыми "клонами", dBXL, Clipper или Foxbase. Если пользователь заинтересован в сохранении своих затрат на эти продукты, необходимо выбрать сервер, который поддерживает данную СУБД. Необходимо также проанализировать вопросы переносимости и производительности. У таких продуктов как Paradox (set-oriented — ориентиро-

ванных на группы записей), возможно, в среде сервера базы данных сохранится высокая производительность. В то же время, у продуктов, подобных dBASE (record-at-a-time — ориентированных на отдельные записи) характеристики производительности могут ухудшиться.

В настоящее время разрабатывается много новых внешних пакетов, имеющих развитый графический интерфейс. Повышение производительности программирования в них достигается за счет использования объектно-ориентированной техники и языка SQL для увеличения продуктивности программирования в многооконной среде. Программные продукты Object/1, Application Manager, Access SQL и SQL Windows открывают возможности разработки оконных приложений для широкого круга SQL-серверов баз данных.

Л.Проничева



По материалам:

1. K. Watterson ServerTechnology: Divide to Conguer. Data Based Advisor. March 1989. p.92-100.
2. DBMS Magazine´s Database Buyer´s Guide and Directory. Vol. 3. Number 6. 1990. p.43-54.
3. Datamation´s Product Guide. Datamation. October 15. 1990. p.84-86.
4. J.Buzzard. SQL Show: The First Annual Gupta Developers Conference. Data Based Advisor. November 1990. p.115-116.
5. J.Moad. What IBM Says About Client/Server. Datamation. February 1 1991. p.54-57.

В первый день выставки Comdex фирма Apple произвела большой фурор, продемонстрировав новые серии рабочих станций Quadra 700 и 900, новое поколение портативных машин и новый Mac Classic II.

Для поддержки новых машин была выпущена следующая версия ОС — v.7.01. Она требует 2 Мбайта ОЗУ и жесткий диск. Фирма заявляет, что новая версия — это не замена System 7.0, а просто ее модификация для новых машин.

Настольная станция Quadra 700 выпускается в четырех конфигурациях — одна только с флоппи-диском, остальные с винчестерами разной емкости. Quadra 900 в корпусе типа "tower" также имеет одну бездисковую конфигурацию и две дисковых. Цены на станции лежат в диапазоне 3,499-9,199 долларов.

Обе системы выполнены на процессоре 68040, работают втрое быстрее Mac IIfx, имеют улучшенный цветной монитор, поддержку Ethernet и новую технологию контроллеров диска SCSI/Nubus. Кроме этого, напольная Quadra 900 может иметь до 64 Мбайт оперативной памяти, дополнительные жесткие диски или другие устройства, подключаемые к интерфейсу SCSI.

Пользователи Macintosh IIcx и IIci могут купить небольшую логическую схему, которая монтируется на плате и превращает их компьютер в станцию Quadra 700. Каждая такая плата имеет 4 Мбайт оперативной памяти и 512 Кбайт памяти с батарейкой.

Новые портативные машины — Powerbook 100, 140 и 170 — поставляются с 2 Мбайтами ОЗУ (расширяется до 8 Мбайт) и 20- либо 40-мегабайтным жестким диском. Цена находится в диапазоне от 2,299 до 4,599 долларов.

Новый Mac Classic II превосходит старый Classic в производительности и поддержке виртуальной памяти благодаря использованию процессора 68030 с тактовой частотой 16 МГц, который вдвое быстрее использовавшегося ранее. Еще одной добавкой к Classic является возможность ввода звукового сигнала.

Newsbytes News Network, 21 October 1991



Это заключительная часть статьи по системам мультимедиа (см. КомпьютерПресс №№ 7,8'91), посвященная перспективам этой технологии в нашей стране.

Мультимедиа - синтез трех стихий

А что у нас?

Теперь, ознакомившись с положением дел в области мультимедиа за рубежом, вернемся домой и оглянемся вокруг. Мы увидим, что системы мультимедиа, этот качественно новый инструмент познания, обучения, общения, которым уже 4 года пользуются школьники Англии, который быстро внедряется во все сферы деятельности и в быт в цивилизованном мире, — у нас остается практически неизвестным, по каким-то непонятным причинам выпав из поля зрения и средств массовой информации, и разнообразных ведомств. Ведомства занимаются компьютеризацией школ, пытаясь оснащать их — в 1991 году! — устаревшими еще до рождения КУВТ-86 и Корветами, способными, в силу своей ненадежности и "недружественности", привить лишь отвращение и страх перед компьютерами. Между тем в мире уже выросло "информатизированное" поколение, для которого сети общего пользования, системы типа телетекста и видеоданных, электронная почта и компьютеры стали естественным, непременным атрибутом окружающей жизни как в профессиональной деятельности, так и дома. Это не кажется страшным, если рассматривать отдельно взятого нашего человека: если "окунуть" его в эту среду, он, быстрее ли, медленнее ли, перестроит стиль мышления, научится управляться с этими информационными инструментами (для ребенка же вообще не будет проблем), однако в масштабах страны мы видим качественное отставание всего общества, всех его институтов, и это становится непоправимым. Появление и повсеместное внедрение систем мультимедиа увеличивает скорость, с которой уходит от нас цивилизованный мир.

Попробуем оценить перспективы, которые могут ожидать мультимедиа в нашей стране. Для этого посмотрим, как обстояли дела в последние годы с внедрением других информационных технологий. Сразу оговорюсь, что все изложенное ниже — личные измышления автора, основанные, однако, на более чем пятнадцатилетнем опыте барахтанья в неспешном потоке развития отечественной вычислительной техники. Безусловно, постоянное сопоставление состояния дел "у нас" и "у них" не может не породить некоторого комплекса вечного несоответствия мировому уровню, опаздывания, повторного изобретения давно изобретенного, порой ощущения полной ненужности и бессмысленности всех усилий. Утешала и поддерживала лишь аналогия с еще одной областью, поневоле близкой — стоматологией, отставшей, пожалуй, еще больше и безнадежней. Тем не менее, осознавая это отставание, наши стоматологи (слава богу!) продолжали сверлить, пилить, лепить, придерживая заветный несточенный бор или щепотку цемента,

привезенную кем-нибудь из-за бугра, для особо доверенных клиентов.

Грустная история

Самый свежий пример — это история отечественных персональных компьютеров. Большая ее часть происходила уже после объявления перестройки и ускорения, и сначала казалось, что ну уж теперь-то!.. По рукам ходили какие-то отксеренные аналитические записки и докладные из высоких сфер: массовые закупки комплектующих и периферии, закупки заводов, совместные производства... И что же? Истинная компьютеризация страны происходит лишь сейчас, с помощью бартерных закупок и сборочных СП, по совсем не персональным ценам и не благодаря, а вопреки усилиям министерств, которые смогли в результате многолетних и дорогостоящих усилий наводнить кладовки, пространства на шкафах и под столами в тысячах учреждений и школ по всей стране пирамидами грязно-серо-черных неработающих, полуработающих, несовсемработающих — зато "чисто самодельных" Электроник, ДВК, Корветов, УКНЦ, Агатов и т.п. Как будто об этой "технике" писал анонимный адресант Генри Баскервиля: "Если вам дороги ваш рассудок и ваша жизнь, держитесь подальше от торфяных болот..."

Подобное положение и с внедрением сетей: министерствам связи европейских стран хватило в конце 70-х годов по 3-4 года от первоначального проекта до введения в полномасштабную эксплуатацию общенациональных сетей общего пользования. Появившиеся примерно тогда же, не без оглядки на Запад, проекты ОГСПД и Академсети, записанные во множество планов и постановлений и разрабатывавшиеся целыми институтами, так где-то и растворились в процессе реализации. Тут, безусловно, значительная часть проблемы заключалась в отсутствии каналов связи, но это лишь подтверждает закономерность: нам при наличии супер-министерства связи (и еще МПСС в недавнем прошлом) осталось немного подождать, пока какое-нибудь из СП не обеспечит Москву, а потом и страну качественной сетью скоростных связных трактов (такие проекты уже есть), другое поставит узлы и необходимый софтвер — и сеть готова. Сейчас такой процесс быстро проходит на территории бывшей ГДР — и она включается в мировую "инфосферу".

Пожалуй, еще более показательна ситуация с локальными сетями: для них не нужны ни дорогие тракты — достаточно мотка дешевого кабеля или даже скрученной пары проводов, ни высокая электронная технология, ни даже глубокие научные разработки: стандарты и протоколы давно опубликованы, прокомментированы, описаны методы их реализации и возникающие проблемы; казалось бы, есть все, чтобы организовать выпуск несложных сетевых контроллеров, реализующих протоколы нижних уровней (для разных типов ЭВМ), приложить программные модули для верхних — и получить недорогую локальную сеть массового применения. Однако на практике в течение многих лет многие умельцы (без иронии!) во множестве контор параллельно создавали уникальные невоспроизводимые локальные сети из подручных средств — вплоть до крейтов КАМАК и шкафообразных МПД ЕС ЭВМ. Это приводило к колоссальным затратам сил, средств, причем часто впустую: очень многие разработки заглохли на разных этапах реализации в связи с непосильностью задачи, слишком большим временем разработки, отсутствием программного обеспечения или просто уходом основного умельца. Массовое же внедрение локальных сетей происходит лишь после снятия эмбарго, путем поставок из-за рубежа готовых комплектов сетей.

Подобная ситуация может повториться и в разработке средств мультимедиа. В связи с тем, что практика уже требует внедрения каких-то элементов мультимедиа, а отечественная промышленность пока ничего предложить не может, вполне вероятна разработка и изготовление "на коленке" единичных плат под конкретные применения.

Итак, мы видим, что опыт развития вычислительной техники в нашей стране на протяжении 80-х годов показывает, что большинство программ проваливаются или реализуются недопустимо медленно при "изоляционистском" подходе к их выполнению. Исходя из этой закономерности, а также задавшись целью обеспечить возможность более быстрого внедрения мультимедиа и других новых технологий в СССР, а не многолетнее спокойное финансирование исследований их развития за рубежом, необходимо максимально ориентироваться на всевозможные совместные формы исследований, разработки и производства.

Ближайшие перспективы

Рассмотрим различные стороны деятельности по исследованию и развитию систем мультимедиа.

1. Работы по отдельным теоретическим аспектам и отдельным компонентам мультимедиа. Они уже сравнительно давно проводятся в стране — это такие направления, как технология гипертекста, базы данных, обработка изображений, звука, комплексные инструментальные системы, графические и анимационные средства. Существуют некоторые наработки и опыт работы в этих областях. Однако они разрознены, их приложение к "истинным" системам мультимедиа, интеграция в единую систему мультимедиа затруднены как отсутствием адекватных технических средств, так и недостаточной координацией этих исследований именно под углом зрения мультимедиа. Тем не менее деятельность по разработке технических и программных средств синхронизации, захвата и обработки изображения, работы со звуковыми данными, стыковки компьютера с аналоговыми устройствами весьма перспективна в наших условиях. Спрос на подобные средства уже велик, отечественный рынок же почти пуст. Причем на доступ-

ной элементной базе наши разработчики и программисты вполне могут создавать продукты, не уступающие зарубежным аналогам. Один из примеров — разработанный малым предприятием "Софт-Хард Технология" цветной видеодигитайзер для IBM-совместимых компьютеров, работающий в реальном времени и обеспечивающий разрешение 832x600 пикселов при 19 битах под цвет пиксела. Плата видеодигитайзера поставляется с пакетом программной поддержки, обеспечивающим работу с ней из языков Си и Паскаль.

2. Разработка и реализация алгоритмов сжатия информации. Основной прогресс в развитии цифровых систем мультимедиа в последнее время обеспечен развитием и реализацией алгоритмов сжатия. В части разработки алгоритмов дела в СССР могут оказаться неплохи — методы сжатия изображений наверняка применялись в космической и военной технике и сейчас, в рамках конверсии, они могут быть приоткрыты и запатентованы; кроме того, кодировщики и математики могут разработать новые алгоритмы или заняться улучшением, дальнейшим усовершенствованием публикуемых алгоритмов JPEG и MPEG. Проблемы возникнут, однако, при реализации разработанных алгоритмов. Здесь сильно мешают отсутствие необходимой технологии и инерционность промышленности. Нам не под силу, подобно маленькой фирме C-Cube Microsystem, разработать СБИС-микросхему, реализующую алгоритм сжатия и сравнимую по сложности с 80386, наладить ее промышленный выпуск, разработать и также запустить в производство плату расширения на основе этой микросхемы, "одеть" все это софтвером и в результате менее чем за год (!) получить общедоступный коммерческий продукт, который любой пользователь может установить на своем компьютере. Можно рассчитывать только на не столь эффективные программные реализации, в лучшем случае — малосерийные платы на дискретных элементах.

   Если думать о перспективе выпуска плат мультимедиа у нас в стране, то возможен вариант частичной реализации алгоритма сжатия на кристалле, чтобы достичь уровня сложности схемы, посильной для нашей технологии, остальная часть алгоритма реализуется программно, возможно, с занесением в ROM. Другой путь — использование RISC-процессоров (которых, впрочем, тоже пока нет — но они по уровню сложности доступны нашей МЭП-технологии). Лучший, быстрейший путь — это некая форма совместной деятельности с цивилизованной страной — от закупки чужих чипов сжатия (их цена сейчас порядка 50-200 долларов) и производства на их основе собственных плат до патентования своих алгоритмов и выпуска реализующих их чипов за рубежом.

3. Выбор политики в области технических средств.

   На первом этапе, этапе исследований и опытного внедрения мультимедиа, единственное приемлемое решение — закупка плат и устройств мультимедиа за рубежом. Здесь приходится задуматься над выбором: какого стандарта мультимедиа следует придерживаться.

   Для начала приходится с сожалением признать, что в ближайшей перспективе выбор массового компьютера практически сделан за нас — это линия IBM PC. Далее, среди многочисленных систем мультимедиа для машин этой линии на начало 1991 года наибольшего внимания заслуживали два варианта: DVI и новые продукты UVC. Однако сейчас все большее количество фирм объявляет о своей поддержке алгоритмов JPEG и MPEG. Наиболее желателен вариант приобретения 3-4 вариантов плат мультимедиа, а испытания проводить при реализации пунктов 4 и 5. В дальнейшем необходимо постоянно следить за мировым рынком мультимедиа, за технической политикой ведущих фирм; при появлении перспективных (или общепризнанных) систем следует обеспечивать их закупку или получение на испытания.

   Тщательный, продуманный выбор между существующими в мире стандартами и направлениями очень важен, чтобы не совершить ошибки, подобной выбору системы SECAM в телевидении, не уйти в тупиковую ветвь, не создать еще один информационный занавес от цивилизованного мира и иметь возможность на выбранных технических средствах создавать продукты, имеющие спрос на мировом рынке.

   Не следует, однако, путать разные задачи. Сказанное выше относится главным образом к "массовому" применению мультимедиа — в образовании и учрежденческой деятельности. В то же время авторские программно-аппаратные комплексы для разработки CD-ROM — электронных книг, руководств, других продуктов гипермедиа чаще создаются на базе Macintosh. Хорош этот компьютер и при подготовке, производстве и использовании интерактивных видеодисков благодаря стекам Hypercard, позволяющим хранить ссылки на конкретные кадры аналогового видеоустройства, гибкой системе управления презентацией пакета Director, а также настольной системе записи и воспроизведения таких дисков, управляемой с Macintosh. Сейчас — наконец-то! — фирма Apple выходит на наш рынок. Будем надеяться, что ей удастся нарушить сложившееся в стране необъяснимое, иррациональное засилье компьютеров линии IBM во всех сферах применения — даже в тех, где они значительно проигрывают своим конкурентам, и что мультимедиа-системы на базе Macintosh станут одним из основных направлений этого наступления Apple.

   Для систем настольных видеостудий может больше подойти Amiga либо, опять-таки, Mac. Для более сложных применений, с жесткими временными требованиями и большим объемом вычислений, систему придется создавать на базе мощных рабочих станций типа Sun или R-6000 IBM.

4. Разработка инструментальных средств мультимедиа, авторских систем. Уже отмечалось, что в этой обла-

сти требуются усилия разработчиков. Однако они могут быть успешными, конкурентоспособными лишь в случае быстрого приобретения новых технических средств, для которых еще не сложился круг привычного для пользователя инструментального софтвера; здесь можно пойти на прямой контакт с фирмой — производителем технических средств. Другая возможная "ниша" — открыть какую-то новую сферу применения имеющихся технических средств и создать специализированную авторскую систему для этой сферы.

5. Разработка конечных продуктов мультимедиа разных типов, а также методик их проектирования и изготовления. Это совершенно новый вид деятельности, сочетающий черты программирования, разработки игр, производства фильмов и др.; в СССР пока практически нет специалистов в этой области. Помимо сложных задач интерактивного программирования, управления информацией, навигации, возникают также проблемы: создания сценария, производства видео-, фото- и аудио-материалов. Как правило, необходимо участие специалистов в предметной области, к которой относится создаваемый продукт.

Некоторый опыт такой работы накоплен в Третьяковской галерее, Русском музее, где уже в течение ряда лет ведутся работы (совместно с зарубежными фирмами) по занесению фондов на интерактивные видеодиски.

Задачи 4 и *5* тесно взаимосвязаны: без опыта производства продуктов мультимедиа трудно оценить слабости существующего авторского софтвера, прочувствовать тонкие места при работе с данной аппаратурой; создаваемую авторскую систему полезно испытывать и оценивать на разработке конкретного продукта. С другой стороны, иногда в процессе разработки большого продукта мультимедиа создается столько инструментальных и вспомогательных программ, что их выпускают на рынок в качестве самостоятельного продукта. Так, в частности, поступила Britannica Software, которая вслед за выходом Compton Multi-Media Enciclopaedia выпустила Publisher CD-ROM Toolkit.

Работы по этим двум пунктам должны привести к появлению и внедрению первых отечественных продуктов мультимедиа. Однако здесь возникает серьезная проблема.

6. Распространение и применение продуктов мультимедиа. Это — вопрос, наиболее критичный в СССР. Дело в том, что сегодня они просто не могут дойти от изготовителя до потребителя. В СССР нет ни CD-ROM, ни оптических стираемых дисков, ни сменных винчестеров, ни ленточных устройств большой емкости, ни скоростных сетей. На дискету 1.44 Мбайт помещается около 10 секунд видео в сжатом формате DVI; передача такого же количества данных через телефонную сеть при очень оптимистической скорости 4800 бод потребует полчаса; выделенные каналы побыстрее в 2-4 раза. Локальная сеть позволит передавать данные мультимедиа лишь в пределах одного учреждения. Полувыходы, первыми приходящие на ум: установка продукта на винчестер пользователя разработчиком с помощью доставки имеющегося у разработчика внешнего накопителя с контроллером и подключения его к машине пользователя; путешествие разработчика со своим компьютером и локальной сетью к пользователю или наоборот; перекачка данных мультимедиа на большую машину и запись на ее ленты, а затем распространение этих лент; при этом, однако, у пользователя также должна быть в досягаемости связка ПК — большой компьютер.

Что касается производства внешних устройств большой памяти, то здесь надеяться на нашу промышленность, видимо, не приходится: за все годы развития ЕС ЭВМ ей (как, впрочем, и болгарской) не удалось создать (скопировать) ни одного дискового магнитного устройства достаточного качества и надежности. Изготовление же оптических устройств, даже при допущении наших передовых позиций в лазерных делах, все равно упрется в больные проблемы точной механики и малообъемной высокочистой химической технологии.

Впрочем, я с радостью готов взять свои слова обратно, если кто-нибудь меня опровергнет. Так, ко мне в последний момент попала очень обнадеживающая бумага — рекламный листочек МНПП "КОМПАКТ" (г. Москва), объявляющий о выпуске этим предприятием и А/О "СИНКВЭС" (Таллинн) отечественных дисководов CD-ROM, полностью соответствующих международным стандартам, подключаемых к IBM PC, работающих с Microsoft CD-Extention и даже воспроизводящих аудиодиски. Постараемся получить более подробную информацию об этой системе, возможно, испытать ее в действии — и о результатах сообщим читателям.

Приобретение для пользователей дисководов CD-ROM могло бы стать хорошим выходом, но при этом придется отсылать подготовленные продукты для тиражирования за рубеж, что достаточно накладно, особенно при малых сериях. Видимо, следует проявлять активность в направлении организации выпуска CD-ROM дисков в нашей стране; такое производство может быть, в частности, организовано на базе фирмы "Мелодия", уже выпустившей первые лазерные аудиодиски. Возможно, какое-нибудь крупное издательство проявит интерес к этому очень перспективному направлению и создаст, например, СП по выпуску оптических и CD-ROM дисков? Напомним, что рынок таких электронных изданий за рубежом достигает миллиардных отметок.

(Здесь я должен извиниться перед читателями за допущенную мной в КомпьютерПресс 7´91 неточность: чистый диск CD-ROM стоит несколько долларов, а чистый видеодиск — порядка 100-200 долларов.)

Каковы же ближайшие практические перспективы разработчика конечных продуктов мультимедиа? Видимо, сначала придется ориентироваться на хорошо оснащенных пользователей, например показательные

"пилотные" школы или учебные центры, или на единичные применения типа тренажеров, "информационных киосков" в музеях и т.п. Кроме того, следует обеспечить такие конфигурации закупаемых учебных комплексов, которые в будущем могли бы легко быть расширены в станции мультимедиа — установкой дополнительных плат или подключением внешних устройств либо даже просто добавлением программ.

7. Наконец, главное в научном плане: исследование мультимедиа как составной части информационных технологий будущего, в комплексе всех связанных проблем; именно здесь существуют наилучшие возможности не копирования зарубежного опыта, а получения оригинальных научных результатов, развития своей линии в области мультимедиа. Однако для этого требуется закупка комплекса технических и программных средств для создания стартовых условий, соответствующих передовому сегодняшнему уровню.

Практические работы с системами мультимедиа (п.п. 4,5) затронут предметы "более высокой теории" — построения ММ-баз, систем гипертекста и гиперизображений, объектно-ориентированных сред и т.п., а также прикладные вопросы отдельных "медий": быстрая компьютерная графика и анимация, настольные видеостудии, компьютер и музыка, электронная и видеопочта, что с необходимостью приведет к активизации работ в этих крайне интересных направлениях. Однако необходимо заниматься и вопросами развития информационных систем в целом, их положения и роли в современном мире и в преобразовании этого мира. Для разработки этих фундаментальных вопросов было бы полезно отойти от IBM PC с их беспорядочно, бессистемно наращиваемым нагромождением хардвера и софтвера (например: DOS, над ним Windows, над Windows ToolBook, над ToolBook перенесенный с Mac Guide, над Guide собственно продукт), и обратиться к компьютеру NeXT, который в наибольшей степени соответствует требованиям времени, поскольку задумывался и создавался именно как информационный инструмент завтрашнего дня, интегрируя последние достижения и теории, и технологии.

О координации

К сожалению, сведения об отечественных работах по мультимедиа получить труднее, чем о зарубежных. В общедоступной компьютерной периодике практически ничего не публикуется, специального издания нет. Поэтому пока исследователи в этой области довольно разобщены.

Большие надежды мы возлагали на межотраслевую конференцию "Оптические дисковые информационные системы", которую должны были провести в октябре 1991 года Минрадиопром и НИИ бытовой радиоэлектронной аппаратуры (г. Львов) — с очень интересным кругом рассматриваемых проблем. Она могла бы послужить началом объединения усилий разработчиков мультимедиа в СССР. Однако — увы — перенесена на неопределенный срок.

Сведения о ряде конкретных отечественных разработок, дошедшие в виде слухов и частных сообщений, нуждаются в проверке, поэтому о них — в следующих номерах.

В настоящее время предпринимаются усилия по созданию Центра мультимедиа. Задачами Центра, помимо проведения собственных исследований в различных областях мультимедиа, являлись бы координация работ с другими организациями в области мультимедиа, вовлечение их в работы по перечисленным выше пунктам, пропаганда технологии мультимедиа, обучение и консультации по использованию имеющихся систем, инициирование исследований во вновь появляющихся перспективных направлениях. Возможно также издание бюллетеня или журнала по мультимедиа и близким тематикам.

Будем надеяться, что темпы проникновения мультимедиа в нашу страну увеличатся — и за счет появления собственных разработок, и за счет более активного выхода на наш рынок зарубежных фирм. Сейчас же либо за рубежом считают, что нам пока за глаза достаточно конторских IBM PC, либо какие-то элементы мультимедиа находятся под контролем КОКОМ как элементы новейших технологий — во всяком случае, на проходящие в стране выставки образцы мультимедиа-продукции почти не попадают. Поэтому пока приходится, довольствоваться сведениями из зарубежной периодики, пытаться получить понятие о целом на основе чьих-то обрывочных впечатлений, иными словами "ощупывать хвост слона", а затем убеждать всех, что слон длинный и тонкий как змея — что я и постарался сделать в этом обзоре.

С.Новосельцев
Сетевой адрес
next@iplan15.iplan.msk.su

Microsoft сделала русский MS-DOS 5.0. Это третий русскоязычный продукт, выпущенный фирмой.

Система, по утверждению разработчиков, работает лучше предыдущей версии. В ней используется переработанный пользовательский интерфейс, многочисленные подсказки — все на русском языке.

Кроме того, система имеет встроенные утилиты восстановления удаленного файла и почти многозадачный режим. Существенно улучшена поддержка принтеров, не имеющих прошитых русских букв. Система будет поддерживать как русский, так и украинский и белорусский языки.

Продаваться русский MS-DOS будет только в комплекте с компьютерами. Это политика фирмы, которая хочет получить хоть какие-то гарантированные прибыли на советском рынке.

А еще у фирмы появился новый дистрибьютор в Киеве — НПП Гамма, имеющий офис на Крещатике.

Сейчас Steepler Ltd. из Москвы в экспериментальном порядке (до конца года) продает русскоязычный MS Works за рубли — 9500 рублей.

Newsbytes News Network, 7 October 1991



Между прочим

Ventura и PaintBrush: маленькие хитрости

Графический редактор PaintBrush IV Plus позволяет решить некоторые проблемы, возникающие при работе с пакетом Ventura Publisher 2.0.

При распечатке вентуровской графики (рамки, скругленные рамки и эллипсы) с серым фоном на принтерах, эмулирующих LaserJet (проверено на C.ITON LIPS 10 и на RICON PCLASER 6000), нередко возникают светлые или темные полосы вместо равномерного тона. Избавиться от этого дефекта можно, заменив вентуровскую графику, созданную в редакторе PaintBrush IV Plus с использованием другого способа представления полутонов серого цвета (способа образования растра). Суть этих замен и превращений в том, чтобы получить возможность выбирать сам способ, частоту растра (LPI) и яркость по своему усмотрению. Для работы, кроме Ventura Publisher, нужен PaintBrush в двух конфигурациях — на 16 цветов и черно-белая.

Подробная последовательность действий такова:

- создать в "вентуре" на чистой странице графический элемент без фона (или скопировать на чистую страницу элемент со сверстанной страницы и выключить фон);
- распечатать страницу с этим графическим элементом в файл. При этом в меню "Сервис"/"Параметры принтера" должен быть выбран принтер LaserJet +, 300dpi и вывод на "имя файла". Получившийся файл печати имеет расширение .COO.
- скопировать файл печати имя файла .COO в каталог пакета PaintBrush IV Plus и конвертировать этот файл в формат .PCX утилитой HP2PCX.EXE. для этого следует ввести командную строку: HP2PCX.EXE «имя файла .COO»
  В результате будет создан файл «имя файла .PCX» (далее именуемый картинкой).
- вызвать цветной PaintBrush и загрузить в него созданную картинку. При загрузке файла выбрать "Convert to color".

Если окажется, что картинка имеет синий фон, то его следует превратить в белый. Для этого надо установить первичный цвет (primary color) синим, вторичный цвет (secondary color) белым, и, перемещая рабочее поле по полю картинки, дважды нажимать color eraser (пятый слева сверху инструмент на инструментальной панели).

- выбрать в меню "Display" опцию "Solid colors only" и залить желаемую область на картинке каким-либо цветом. Цвета соответствуют полутонам серого по увеличению черноты в следующем порядке:
  белый, желтый, светло-голубой, светло-серый, светло-зеленый, серо-голубой, светло-фиолетовый, коричневый, темно-зеленый, темно-серый, красный, темно-фиолетовый, темно-красный, синий, темно-синий, черный.

Впрочем, для некоторых цветов разница между соответствующим им полутонам серого трудно различима. После заливки нужно сохранить картинку, проверить ее размеры ("Edit" "Change entire image" "Set untis") и покинуть цветной PaintBrush;

- вызвать черно-белый PaintBrush и загрузить в него созданную цветную картинку. При загрузке картинки выбрать "Set halftones" и в появившемся диалоговом окне выбрать тип, яркость и частоту растра.

Опыт подсказывает, что тип растровой сетки (halftone pattern) "Fatting" меньше всего подвержен дефектам при распечатке. Отпечатанная картинка при использовании этого растра напоминает обычную газетную фотографию (для разрешения 300x300 точек на дюйм). Если использовать как фон графики, то получается очень похоже на фон обычного окна в "вентуре". Тип "Diffused" выглядит довольно привлекательно — своим рисунком он напоминает тисненую бумагу, обычно более подвержен искажениям при распечатке, чем "Fatting". Тип "Enhanced" похож на "Diffused", но рисунок его тоньше, а подверженность искажениям еще больше. Типы "Bayer" и "Hex bayer" очень похожи на фон "вентуровской" графики и столь же подвержены искажениям.

Установка яркости (brightness) равной 128 или немного больше дает вполне приличный результат. Установка частоты растра (halftone screen size) 75 LPI (line per inch —линий на дюйм) обычно тоже дает приличный результат. Рост величины этого параметра делает рисунок растра тоньше, но и чувствительнее к искажениям при печати, а градации серого — менее различными. Однако изменения этого параметра влияют только на "Fatting";

- после того, как цветная картинка желаемым образом конвертирована в черно-белую, обязательно следует сохранить ее и покинуть редактор. В случае выхода без сохранения результаты конверсии будут потеряны;
- теперь у нас есть черно-белая картинка, размеры ее нам известны. Остается вновь загрузить Venture, создать на месте графического окна обычное окно нужного размера и загрузить в него нашу картинку.

Эта же технология может быть полезна, при использовании для иллюстрирования издания копий экрана, "захваченных" утилитой freeze.exe. Если такую копию загрузить в Venture без предварительной обработки, то одни цвета превратятся в черный, другие —в белый, в результате чего часть изображения станет неразличимой. Если же конвертировать эти цвета в полутона серого, то у Вас получится достаточно приличная иллюстрация.

В. Каминский

НОВОСТИ

Компьютерные взломщики все еще правят миром — так быть не должно

Корреспонденты Newsbytes Джон и Барбара Макмуллен сообщают, что американские компьютерные системы все еще очень уязвимы для нападения хакеров. Эммануэль Голдстейн, издатель 2600 Magazine: The Hacker Quarterly, заявил в Нью-Йорке, что "американская общественность часто дает себя убаюкать ложному чувству безопасности", для которого факты компьютерного взлома не дают никаких оснований. Он продемонстрировал записи того, как голландский хакер взламывал компьютеры армии США, а также показал, с какой легкостью можно проникнуть в компьютерные системы американских деловых и правительственных учреждений.

Датский взломщик завел фиктивного пользователя под именем "danquale" (вице-президент США Dan Quale) и затем получил полный контроль над системой. Голдстейн сказал, что то, что важным пользователям дают очевидно угадываемые пароли типа "Kuwait", вызывает серьезную тревогу. По его словам, взлом системы не причинил никакого ущерба. В нескольких случаях физического взлома было отмечено использование замка фирмы Simplex всего лишь с 1,085 возможными комбинациями. Многих ни в чем не повинных людей винят за то, с какой легкостью открывается этот замок.

Голдстейн сделал абсолютно правильное заявление. В течении многих лет американская политика в этом вопросе сводилась к "преследованию" и "жестким" приговорам в отношение небольшого числа преступников, которые были пойманы, вместо использования элементарных методов предотвращения преступлений, которые могли бы оказаться дешевыми, легкими и эффективными. Причины, по которым не сделано то, что нужно, несостоятельны как с моральной, так и с финансовой точки зрения.

The Teleputing Hotline, October 24, 1991

Новые лазеры дают ключ к плоским экранам

Исследователи в американских университетах Purdue и Brown сообщили, что новые сине-зеленые лазеры могут сильно улучшить качество всех плоских дисплеев. Синие лазеры используют меньшую длину волны, чем красные. Их разработка проводилась на коммерческой основе для фирмы 3M. По словам исследователей при использовании этих лазеров на компакт-дисках и CD-ROM (ПЗУ на компакт-дисках) можно будет поместить в 4 раза больше данных. Ключом к созданию синих лазеров стала молекулярная лучевая эпитаксия, при которой ученые создают структуры на уровнях одного атома. Устройства, использующие эту технологию, могут появиться в течение одного года.

The Teleputing Hotline, October 24, 1991

AT&T создает канал связи для домашних фирм

AT&T создала AT&T Home Office Network, первую субсидируемую корпорациями ассоциацию, поддерживающую нужды людей, работающих дома. Чтобы стать членом ассоциации, нужно чтобы ваш телефон был заабонирован в AT&T. Компания планирует взимать плату после одного года работы. Члены ассоциации получают ежеквартальный журнал, скидки на продукцию и услуги AT&T и могут рассчитывать на более низкие цены у таких фирм, как Panasonic, UPS и др. AT&T оценивает число американцев, которые работают дома в своих личных фирмах, в 11.8 миллиона человек, причем ежегодный прирост составляет 10%.

The Teleputing Hotline, October 24, 1991

DISCOVER будет телефонной карточкой для SPRINT

Идя по стопам AT&T, перешедшей на кредитные карточки, и MCI, заключившей союз с Visa, Sprint подписала соглашение, в результате которого кредитная карточка Discover фирмы Sears станет телефонной карточкой для телефонной сети Sprint, связывая базы данных, используемые обоими фирмами. Новая программа, названная Value-Phone, предоставит пользователям Discover некоторые скидки на звонки по сети Sprint, сделанные с помощью карточек.

The Teleputing Hotline, October 24, 1991

MILLIDYNE представляет сотовые модемы CDLC

Millidyne, компания, работающая в паре с Millicom, анонсировала два сотовых модема, поддерживающих протокол передачи Cellular Data Link Control — альтернативу MNP 10 с коррекцией ошибок фирмы Microcom. Протокол Cellular Data Link относится к public domain. CDLC используются сетью Vodafone в Великобритании. Согласно протоколу сначала формируются поля данных, а затем они передаются в кадрах, в которые добавляется поле синхронизации, поле коррекции ошибок, чередование битов и селективная повторная передача. Модемы можно будет приобрести в ноябре по цене 795 долларов за штуку. Они работают на скорости 2400 бод. Millidyne также будет поставлять фирмам, осуществляющим сотовую связь, оборудование для отслеживания передаваемых данных и другие изделия.

The Teleputing Hotline, October 24, 1991

Следует ждать в скором времени появления дешевых модемов со скоростью передачи 14,400 бит в секунду — компания Phylon объявила, что ими создан набор из трех микросхем V.32BIS, который может осуществлять передачу как данных, так и факсимильных сообщений с такой скоростью, а также на всех более низких скоростях согласно международным стандартам CCITT. Новый набор микросхем включает в себя все, что нужно для обработки голосовой почты, регистрации звонков, маршрутизации факсимильных сообщений и идентификации звонящего абонента — даже в компьютере класса laptop. Ранние версии набора использовались в модемах фирмы Hayes. Новые схемы стоят всего 50 долларов при оптовых закупках.

The Teleputing Hotline, October 17, 1991

Несмотря на катастрофическое положение в области прав человека, Китай вызывает растущий интерес со стороны американских фирм, связанных с телекоммуникацией. Недавно фирма Hayes, производитель модемов, расширила свое производство, а Sprint объявила о контрактах, согласно которым в стране появятся ее служба передачи сообщений Telemail и службы SprintMail X.400. Sprint сделала это объявление одновременно с сообщением о заключении аналогичного соглашения с Тайванем. Тем временем агентство ЮПИ сообщило, что семь человек расстре-

ляно в Китае за кражу медного кабеля. Государственная газета Legal Daily назвала такие кражи "сильно распространенными".

The Teleputing Hotline, October 17, 1991

Семь региональных компаний Bell хотят, чтобы налогоплательщики заплатили им от 300 миллионов до 1 миллиарда долларов, прежде чем они доведут волоконную связь до всех уголков страны. Таково заключение Рича Тома, президента фирмы Telcommunications Solutions for America, основанное на результатах исследований, проведенных для компаний Bell такими фирмами, как Arthur D. Little и Data Resources.

Тома заявил, что эти субсидии запрашиваются помимо требований Bell об отмене запрета на участие в предоставлении информационных услуг, и сверх их запросов на замену используемой около века схемы "нормы прибыли" на регулирование потолка цен. Субсидии могли бы принять форму налоговых льгот, быстрого списания действующего оборудования или прямой выплаты денег. Недостаток капиталовложений компаний Bell в свои сети угрожает общей конкурентоспособности США, добавил Тома. "Зачем компаниям приобретать новую технологию, если MCI Mail тащится со своим модемом на 2,400 бод, или их модем на 9,600 все время сваливается на более низкую скорость передачи", — сказал он. "Телефонные сети являются местными сетями, которыми пользуются все, и они находятся в плохом состоянии". Вместо того, чтобы их улучшать, добавил он, Bell вкладывает деньги, полученные от абонентов, за рубежом, приобретая иностранные телефонные компании и концессии на сотовую связь, хотя обещают крупные капиталовложения, чтобы довести свои услуги до уровня стандартов XXI века.

The Teleputing Hotline, October 17, 1991

Подписка на русскую версию Teleputing Hotline открылась в сети электронной почты RELCOM.

Teleputing Hotline выходит дважды в неделю (1 выпуск объемом 10 Кбайт), редактируется в Атланте (США) и Лондоне, и стоит для абонентов Релкома всего 85 рублей в месяц в отличие от американской версии, продающейся за $50. Русская версия Hotline выходит в свет через пять часов после появления его за океаном.

На этой странице помещен бланк заказа на журнал «КомпьютерПресс»
Вы можете его вырезать и, заполнив, отправить нам по адресу:
113093, Москва, а/я 37

Подписка на 1992 г. принимается до 31 января 1992 г. Число экземпляров — без ограничений.

Стоимость годовой подписки на "КомпьютерПресс" — 57 рублей 60 копеек.

Деньги следует перечислить на расчетный счет агентства "КомпьютерПресс".

Наши банковские реквизиты:

получатель: Автобанк (для зачисления на счет №345708)
расчетный счет получателя: №161202
банк получателя: ЦОУ при Госбанке СССР. МФО №299112.

Бланк заказа и заверенную копию платежного поручения или почтовую квитанцию о переводе денег на счет Агентства следует приложить к заказу.

Без одновременной оплаты подписной стоимости заказ не принимается.

ЗАКАЗ

От кого ______________________________

Адрес ______________________________

____
(ПОЧТОВЫЙ ИНДЕКС УКАЗЫВАТЬ ОБЯЗАТЕЛЬНО)

Прошу оформить подписку на журнал КомпьютерПресс на 1992 год

Подписная плата в сумме ______________________ перечислена

платежным поручением (почтовым переводом) № ______________

от ______________________ 199__ г.

(Копия платежного документа прилагается).

Заказ

Советско-американское предприятие "Соваминко"

Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 191186, Ленинград, Невский проспект, 28, Магазин № 1 "Дом книги"

От кого . . . .

Адрес . . . .

(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

Заказ

Советско-американское предприятие "Соваминко"

Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 630076, Новосибирск, Красный проспект, 60 Магазин № 7 "Техническая книга"

Телефон для справок: 20-05-09

От кого . . . .

Адрес . . . .

(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

QUATTRO PRO 3.0 — ШЕДЕВР, ДОСТУПНЫЙ ВСЕМ



Вы устали от монотонной работы с цифрами и документами? Вы думаете, что при общении с компьютером необходим посредник-программист? Вы работаете с программными продуктами только надежных, всемирно известных фирм?
Электронная таблица QUATTRO PRO 3.0 американской корпорации orland Internationalпозволит Вам общаться с компьютером на «ты» и возьмет на себя всю тяжесть текущей работы.
Любые бухгалтерские и экономические расчеты, возможность вести всю документацию предприятия, встроенная издательская система и сложные отчеты типографского качества, деловая графика и даже электронные слайдофильмы со звуковыми эффектами — завтра Вы не будете мыслить свою жизнь без QUATTRO PRO. Спектр технических характеристик QUATTRO PRO удивительно сочетается с неприхотливостью программы: даже на IBM PC/XT можно работать в графическом режиме, использовать автоувеличение фрагментов таблицы от 25 до 100%, печатать во всех мыслимых и немыслимых режимах, включая бесконечные графики на непрерывной ленте, подбирать оптимальную для глаз палитру красок и формировать содержимое меню.
Стоимость пакета QUATTRO PRO — 4000 рублей или 495 долларов США.
Общение с QUATTRO PRO приносит радость — приходите, и мы убедим Вас в этом.

СП «Интерквадро» Москва, 2-й Новоподмосковный пер., 4. Телефон 150-92-01

B O R L A N D

Makers of Paradox,® Quattro® Pro, ObjectVision,™ Borland® C++, Turbo C++, Turbo Pascal® and Sidekick®

*Paradox SQL Link ($495 U.S. suggested retail) is an add-on product for Paradox 3.5 and is required to make the connection with SQL servers. **Microsoft, IBM, SYBASE, Oracle and DEC Rdb/VMS servers supported now. †Offer also good for owners of any version of R:BASE or DataEase. Proof of ownership (a page from the manual) is required. Satisfaction guaranteed or your money back. Upgrade offer expires July 31, 1991. Upgrade pricing good in U.S. and Canada only. Mail orders to: Borland International, Inc., P.O. Box 660001, Scotts Valley, CA 95067-0001. Add $9.00 for shipping and handling. Residents in CA, CT, GA, IL, MA, MI, MN, NJ, NY, OH, PA, TX, VA and WA, please add appropriate sales tax. MI, PA, and TX residents please add tax to freight charges. For orders outside the U.S., call (408) 438-5300. All prices are in United States dollars. BI 1395B

CODE: ME74

Цена 3.15





МАКЛИНКЕР УНИВЕРСАЛЬНЫЙ

автоматическое устройство для прокраса и восстановления лент матричных принтеров шириной до 24 мм. Обеспечивает 100% восстановление красящих свойств, что позволяет Вам продлить срок эксплуатации одной ленты в 30-50 раз при условии своевременного прокраса. В комплект поставки входит 1 литр тонера, обеспечивающий прокрас 2000 м 13-ти миллиметровой ленты. Поставка дополнительного тонера по желанию заказчика.

"ВЕСНА-4"

многофункциональное интеллектуальное устройство на базе однокристальной микроЭВМ — обеспечивает мгновенное определение номера звонящего абонента, а также предлагает пользователю множество сервисных удобств, вплоть до охранной квартирной сигнализации.



МОДЕМ COMPLINK-C4800

для IBM PC AT/XT, обеспечивает передачу информации на отечественных телефонных линиях со скоростью не менее 1200 бит/с на выделенных линиях — не менее 4800 бит/с.

Фирма КОМП — единственный в стране разработчик и производитель представленной продукции. Обеспечивает немедленную поставку и гарантийное обслуживание.

Телефон: (095) 353-68-58
с 11 до 18 часов



Research centre COMP LTD.

109388, h.28, Shosseinaya st., Moscow, USSR.